robotron

Programmtechnische
Beschreibung

REDABAS

Relationales
Datenbanksystem

H 0313-2053-1 M 0030

Arbeitsplatzcomputer A 7100
Bürocomputer A 5110, A 5120, A 5130
Personalcomputer robotron 1715

Betriebssystem
SCP 1700

SYSTEMUNTERLAGEN DOKUMENTATION Stand 05/87	Programmtechnische Beschreibung REDABAS	POS SCP SCP1700

Programmtechnische
Beschreibung

POS

R E D A B A S

Realationales
Datenbankbetriebssystem

VEB Robotron-Projekt Dresden

H 0313-2053-1 M 0030

Die vorliegende Systemunterlagen-Dokumentation REDABAS entspricht dem Stand von 5/87

Nachdruck, jegliche Vervielfaeltigung oder Auszuege daraus sind unzulaessig

Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Projekt Dresden.

Im Interesse einer staendigen Weiterentwicklung werden alle Leser gebeten, Ihre Vorschlaege bzw. Hinweise dem Herausgeber oder den untengenannten Vertreibern mitzuteilen.

Herausgeber: VEB Robotron-Projekt Dresden
Leningrader Str. 9
Dresden, 8010

Das Programmpaket REDABAS und diese Dokumentation werden vertrieben durch

- VEB Robotron-Bueromaschinenwerk Soemmerda
  Abt. KA

  PSF 43

  Soemmerda

  5230

fuer die Erzeugnisse robotron A 5110 und robotron 1715

- VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt
  Software-Zentrum / HF

  PSF 129

  Karl-Marx-Stadt

  9010

fuer die Erzeugnisse robotron A 5120 / A 5130 und
K9824 / 27 / 31

- VEB Robotron-Projekt Dresden
  Bereich K

  Leningrader Str. 9

  Dresden

  8010

fuer den Arbeitsplatzcomputer A7100

Kurzreferat

Zur Anwenderdokumentation von REDABAS gehoeren die Schriften

- Anwenderbeschreibung
- Programmtechnische Beschreibung

Die Anwenderbeschreibung enthaelt

- eine Kurzcharakteristik von REDABAS sowie einen Ueberblick ueber Leistungsfaehigkeit und moegliche Anwendungsbereiche

- eine einfuehrende Beschreibung der Handhabung von REDABAS mit Beispielen

- Aussagen ueber die Voraussetzung fuer die Anwendung bezueglich Hardware und Software

- eine Kurzbeschreibung der Funktionen der Datenbankverwaltung und der Programmiermoeglichkeiten.

Die Programmtechnische Beschreibung enthaelt detaillierte Angaben ueber die Arbeitsweise, die Funktionen der Datenbankverwaltung, das Programmieren und die Datenorganisation von REDABAS.

Eine ausfuehrliche Beschreibung der REDABAS-Funktionen und -Befehle bietet dem Anwender in Verbindung mit praxisnahen Demonstrationsbeispielen exakte Mittel fuer die Handhabung und Nutzung von REDABAS.

Ausser den beiden genannten Schriften wird noch eine Referenzkarte angeboten.

Inhaltsverzeichnis Seite

Einfuehrung 8

1. Leistungsparameter von REDABAS 10

2. Arbeitsweise mit REDABAS 11
2.1. Voraussetzung fuer die Anwendung 11
2.2. Installieren 11
2.3. Allgemeine Hinweise 13
4. Starten und Beenden 15

3. Funktionen der Datenbankverwaltung 17
3.1. Einrichten und Aendern von Dateien 17
3.1.1. Erstellen einer Datei 17
3.1.2. Eingeben weiterer Datensaetze 22
3.1.3. Aendern von Datensaetzen 23
3.1.4. Loeschen von Datensaetzen 27
3.2. Wiederauffinden aus der Datenbank 29
3.2.1. Selektion bestimmter Datenfelder 29
3.2.2. Bedingte Auflistung von Datensaetzen 30
3.2.3. Bedingte Anzeige von Datensaetzen 36
3.2.4. Positionieren auf Datensaetze 36
3.2.5. Abfragen einzelner Datenfelder 37
3.3. Aendern der Dateistruktur 39
3.4. Sortieren und Indizieren 43
3.5. Auswerten und Verdichten 48
3.5.1. Summenbildung 49
3.5.2. Zaehlen von Datensaetzen 50
3.5.3. Verwendung temporaerer Speichervariablen 50
3.5.4. Verdichten von Dateien 51
3.6. Listengenerator 54
3.7. Gleichzeitiges Arbeiten mit mehreren Dateien 66
3.7.1. Gleichzeitiges Bearbeiten von zwei Dateien mit unterschiedlicher Dateistruktur 66
3.7.2. Verknuepfen von zwei Dateien 72
3.8. Zusammenfassung 74

4. Programmieren 75
4.1. Befehlsfolgen 75
4.2. Programmschleifen 83
4.3. Programmverzweigungen 89
4.4. Programmierte Dateneingabe 95
4.5. Zusammenfassung 99

5. Datenorganisation 100
5.1. Aufbau der Datenbank 100
5.2. Datentypen und Feldformate 101
5.3. REDABAS - Dateien 102
5.3.1. Datenbankdateien (.DBD) 102
5.3.2. Indexdateien (.IDX) 103
5.3.3. Variablendateien (.VAR) 103
5.3.4. Befehlsdateien (.PRG) 103
5.3.5. Reportdateien (.DEF) 104
5.3.6. Maskendateien (.MSK) 104
5.3.7. Textdateien (.TXT) 104
5.3.8. Sicherungsdateien (.BAK) 105

6. Hinweise zur Dialogfuehrung 105
6.1. Fehlerbehandlung 105
6.2. Seitenmodus und Cursorsteuerung 106

7. Ausdruecke 110
7.1. Variablen 110
7.2. Konstanten 110
7.3. Operationen 111
7.3.1. Arithmetische Operationen (Berechnungen) 111
7.3.2. Vergleichsoperationen 112
7.3.3. Logische Operationen 113
7.3.4. String-Operationen 114
7.3.5. Rangfolge von Operationen 114
7.4. Funktionen 115
7.5. Makros 125

8. Befehle 127
8.1. Allgemeines 127
8.2. Symboldefinitionen 129
8.3. Beschreibung der Befehle 132

9. REDABAS-Schnittstellen 185
9.1. Daten-Schnittstelle 185
9.2. Programm-Schnittstelle 186

Fehlermeldungen 187

Abkuerzungungsvereichnis 191

Sachwortverzeichnis 191

Bildverzeichnis

		Seite
Bild 1	Struktur der Datei "kunden"	19
Bild 2	Maske fuer die Datenerfassung	17
Bild 3	Liste der Datei "kunden"	19
Bild 4	Struktur der Datei "kunden"	19
Bild 5	Maske fuer das Anfuegen eines neuen Datensatzes	20
Bild 6	Erweiterte Datei "kunden"	20
Bild 7	(Unveraenderter) Datensatz Nr. 00004	21
Bild 8	Veraenderter Datensatz Nr. 00004	21
Bild 9	Datei "kunden" nach der Aenderung	22
Bild 10	Aendern eines Datensatzes mit "CHANGE"	24
Bild 11	Datei "kunden" mit den zur Loeschung markierten Datensaetzen	25
Bild 12	Drei markierte Saetze der Datei "kunden" sind physisch geloescht	26
Bild 13	Datei "kunden", spezifiziert nach Vorname, Name, Ort	27
Bild 14	Datei "kunden", in zwei Varianten durchsucht	27
Bild 15	Datei "kunden", mit Vergleichsoperator "=" durchsucht	28
Bild 16	Datei "kunden", mit Vergleichsoperator " " durchsucht	29
Bild 17	Demonstration der "Substring"-Funktion	29
Bild 18	Zwei mit "OR" verknuepfte Bedingungen	30
Bild 19	Zwei mit "AND" verknuepfte Bedingungen	34
Bild 20	Datei "kunden", mit positioniertem Suchbegriff durchsucht	31
Bild 21	Beispiel fuer die Anwendung der Befehle LOCATE, DISPLAY, CONTINUE	32
Bild 22	Varianten des Befehls DISPLAY	33
Bild 23	Varianten des Befehls GO	33
Bild 24	Abfragen einzelner Felder mit dem Fragezeichen	34
Bild 25	Maske fuer Aenderung der Dateistruktur	36
Bild 26	Neue Struktur der Datei "kunden"	37
Bild 27	Durch den Befehl BROWSE mit der Kundennummer komplettierte Datei "kunden"	38
Bild 28	Datei "kunden", sortiert nach der Kundennummer	39
Bild 29	Datei "kunden", indiziert nach der Postleitzahl	41
Bild 30	Struktur der Datei "auftrag"	42
Bild 31	Datei "auftrag"	42
Bild 32	Uebersicht der Speichervariablen	45
Bild 33	Zusammenfassung der Auftragsdatei nach Kunden	46
Bild 34	Datei "gesamt", nachdem ueberfluessige Felder eliminiert wurden	46
Bild 35	Struktur der Datei "bestand"	47
Bild 36	Datei "bestand"	48
Bild 37	Dialog fuer die Erstellung einer Liste "Lagerbestand" (Reportdatei "best1")	50
Bild 38	Mit dem Listengenerator erstellter Lagerbestand (Reportdatei "best1")	51
Bild 39	Dialog fuer die Definition der Reportdatei "best21"	59

Bild 40	Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, mit Einzelpositionen Reportdatei "best21")	61
Bild 41	Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, nur Zwischensummen und Summe (Reportdatei "best22")	62
Bild 42	Dialog zur Definition der Liste "Ersatzteiluebersicht" (Reportdatei "best31")	63
Bild 43	Liste "Ersatzteiluebersicht" (Reportdatei "best31")	64
Bild 44	Bedarfsliste	65
Bild 45	Struktur der Datei "aufpos"	66
Bild 46	Datei "aufpos"	66
Bild 47	Befehlsfolge fuer das gleichzeitige Bearbeiten von zwei Dateien mit unterschiedlicher Struktur	69
Bild 48	Datei "aufpos" nach der Bearbeitung	69
Bild 49	Struktur der Datei "rech"	70
Bild 50	Datei "rech"	70
Bild 51	Datei "auftrag" nach der Aktualisierung	72
Bild 52	Mittels JOIN gebildete Datei "kundrech"	73
Bild 53	Aufruf und Abarbeiten von "prog1"	77
Bild 54	Zusammenstellung wesentlicher Informationen ueber einen Kunden	79
Bild 55	"prog2" - Programm zur Erstellung eines Kundenbelegs	83
Bild 56	Programm "rechnung", mit Programmschleife realisiert	84
Bild 57	Programm "progkun1" zur Erstellung von Kundenbelegen, mit Programmschleife realisiert	86
Bild 58	Programm "progkun2" zur Erstellung von Kundenbelegen - mit Programmverzweigung	90
Bild 59	Menueauswahl	92
Bild 60	Menueprogramm "menue1"	93
Bild 61	Programm "aufnahme"	96
Bild 62	"menue2" - komplettes Programm zur Erstellung des Bildschirmmenues	98
Bild 63	Tabelle eines Lagerbestandes	100

Einfuehrung

Die immer staerkere Orientierung auf den Einsatz von Personal- und Arbeitsplatzcomputern fuer einen Anwenderkreis, der kleine und mittlere Datenmengen zu verwalten hat, erfordert die Bereitstellung adaequater Software.

Die Vorteile von Datenbanksystemen fuer eine Vielzahl von Anwendungen sind bewiesen. Neben Leistungsfaehigkeit wird heute von einem modernen Datenbanksystem Uebersichtlichkeit und Einfachheit gefordert, um neben Spezialisten auch dem Nutzerkreis ohne rechentechnische Spezialkenntnisse schnelle und flexible Loesungen seiner Aufgaben zu ermoeglichen.

Dieser Tendenz entspricht der relationale Aufbau eines Datenbanksystems, der gegenueber hierarchischen bzw. vernetzten Datenbanksystemen u.a. folgende Vorteile aufweist:

- einfache Datensicht in Form von Tabellen
- einfache Implementierung der logischen Datenstruktur
- Bestehende Programme werden von Datenbank-aenderungen nicht beeintraechtigt.

Das relationale Datenbankbetriebssystem REDABAS zeichnet sich neben diesen Vorzuegen durch hohe Flexibilitaet im Datenzugriff und die integrierte Kommandosprache aus, die in einem einheitlichen Sprachkonzept Datenverwaltung und Anwendungsprogrammierung gestattet.

REDABAS beinhaltet neben den Grundfunktionen zur Datenbeschreibung und -manipulation das Sortieren nach einem beliebigen Ordnungsbegriff, das Indizieren von Datenbestaenden, das Auswerten nach bestimmten Kriterien, das Summieren von Feldern, das Verknuepfen mehrerer Dateien und eine komfortable Druckaufbereitung.

Der Vorteil des auf dem Relationenkonzept basierenden Datenbankbetriebssystems REDABAS ist seine im Vergleich zu anderen Datenbankbetriebssystemen leicht verstaendliche Nutzersicht in Form von Tabellen. Diese Sicht in Tabellen entspricht der in vielen Anwendungsgebieten vorkommenden Darstellungsform von Informationsbestaenden.

Auf die mathematische Theorie der Relationen uebertragen, stellt eine Tabelle eine Relation dar, bestehend aus einer Anzahl gleichartiger Zeilen, den Tupeln. Im traditionellen Sinn entspricht eine Relation einer Menge gleichartiger Datensaetze.

Die im Beispiel dieser Programmtechnischen Beschreibung angefuehrte Kundendatei mit den Feldern NAME, VORNAME, PLZ, ORT, STRASSE ist z.B. eine Relation.

Die Beziehungen zwischen den Datensaetzen verschiedener Dateien werden durch inhaltliche Bezuege hergestellt, d.h. ueber die Datenfelder zweier Satzarten, die gleiche Feldwerte beinhalten.

So ist im Demonstrationsbeispiel die Beziehung zwischen der Kunden- und der Auftragsdatei ueber die Kundennummer gewaehrleistet.

Das Modifizieren bestehender Relationen (Dateien), das Erstellen und Anfuegen neuer Relationen an eine Datenbank ist ohne Einfluss auf schon vorhandene Datenbestaende moeglich.

Das relationale Konzept ermoeglicht auch die Anwendung einer relationalen Datenbanksprache, mit der, basierend auf der Relationenalgebra, Relationen einfach und mathematisch exakt manipuliert werden koennen. Mit einer Operatorfunktion koennen ganze Tabellen in einem Schritt verarbeitet werden. Daraus leitet sich die Maechtigkeit der REDABAS-Datenbanksprache ab.

Zusaetzlich zu den Befehlen der Datenmanipulation bietet REDABAS im einheitlichen Sprachkonzept Befehle, die eine komplette Anwendungsprogrammierung erlauben. So koennen Datenmanipulationen direkt in die Verarbeitungsprogramme integriert werden, ohne dass bestimmte Konventionen einzuhalten sind.

Nichtprogrammierer oder Anwender, die sich zunaechst einarbeiten wollen, werden als erstes Dateiverwaltungen mit REDABAS vornehmen. Dies ist schnell und problemlos mit der vorliegenden Programmtechnischen Beschreibung moeglich. Nach kurzer Einarbeitung werden diese Nutzer in der Lage sein, mit demselben Befehlsvorrat und einigen zusaetzlichen leicht erlernbaren Befehlen Programme zu schreiben.

Ein weiterer Vorteil, der sich aufgrund des durchgaengigen Sprachkonzepts von REDABAS ergibt, ist die redundanzarme Datenverwaltung in einer Datenbank, da sowohl Nichtprogrammierer fuer ihre Aufgaben der Dateiverwaltung als auch Anwendungsprogrammierer auf dieselben Datenbestaende zugreifen.

1. Leistungsparameter von REDABAS

max. Anzahl von Dateien	beliebig
Anzahl der waehrend der Verarbeitung eroeffneten Dateien	max. 16
Saetze pro Datei	max. 65535
Zeichen pro Satz	max. 1000
Felder pro Satz	max. 32
Zeichen pro Feld	max. 254
Feldformat	fest
Darstellungstyp	alphanumerisch, numerisch, logisch
groesste Zahl	ca. $\pm 1 \cdot 10^{63}$
kleinste Zahl	ca. $\pm 1 \cdot 10^{-63}$
numerische Genauigkeit	10 Ziffern
Zeichenkettenlaenge	max. 254 Zeichen
Zeilenlaenge eines Befehls	max. 254 Zeichen
Listenkopflaenge	max. 254 Zeichen
Indexschluessellaenge	max. 100 Zeichen
Anzahl Speichervariablen	max. 64
Anzahl Ausdruecke im SUM-Befehl	max. 5
Groesse des Variablenbereiches	max. 1536 Zeichen

2. Arbeitsweise mit REDABAS

2.1. Voraussetzung fuer die Anwendung

REDABAS wird sowohl fuer 8-Bit-PC unter dem Betriebssytem SCP als auch fuer 16-Bit_AC unter dem Betriebssystem SCP1700 bereitgestellt
Hardwaremaessig muessen diese Computer mit mindestens 64 K Bytes RAM und einem Diskettenlaufwerk ausgeruestet sein

Das Kombinat Robotron bietet fuer die Arbeit mit REDABAS folgende Computer an:
- unter Steuerung des Betriebssystems SCP
  Buerocomputer der Erzeugnisreihen A 5110, A 5120 und A 5130
  Personalcomputer PC1715
- unter Steuerung des Betriebssystems SCP1700
  Arbeitsplatzcomputer A7100

Die REDABAS-Programmversionen fuer SCP und SCP1700 sind kompatibel. Sowohl die von REDABAS aufgebauten Dateien als auch die durch den Nutzer erstellten REDABAS-Anwndungsprogramme koennen unter beiden Betriebssystemen ohne Konvertierung verarbeitet bzw. ausgefuehrt werden

REDABAS stellt fuer die Datenbankverwaltung und Programmierung eine eigene Datenmanipulationssprache zur Verfuegung, in der die Anweisungen zur Bearbeitung der Datenbank formuliert werden.

Bevor Sie mit REDABAS arbeiten, muessen Sie sich mit dem Befehlsvorrat von REDABAS vertraut machen. Im Abschnitt 3. werden die Funktionen der Datenbankverwaltung vorgestellt, und an einem Demonstrationsbeispiel wird die Anwendung sofort erlaeutert. Fuer das Vertrautsein mit REDABAS und eine rasche Praxiswirksamkeit ist zu empfehlen, dieses Demonstrationsbeispiel auf dem eigenen Personalcomputer nachzuvollziehen.

Es ist auch moeglich, Befehle zu Programmen zusammenzufassen, zu speichern und wunschgemaess aufzurufen. Die Programmierung mit REDABAS wird im Abschnitt 4. vorgestellt.

2.2. Installieren

Fuer die Anpassung von REDABAS an Ihren Rechner mit seiner speziellen Ausstattung steht das Programm RDBASINS bereit.

Bevor Sie jedoch das Installierungsprogramm aufrufen, sollten Sie unbedingt eine Sicherheitskopie der REDABAS-Programmkomponenten anfertigen. Das Installierungsprogramm veraendert die Programmdatei REDABAS.COM (SCP) bzw. REDABAS.CMD (SCP1700), mit der Sicherheitskopie koennen Sie auch weiterhin auf deren urspruenglichen Inhalt zurueckgreifen.

Das Installierungsprogramm umfasst im wesentlichen die zwei

- Anpassung von REDABAS an Ihren Bildschirm und
- Auswahl von Programmparametern.

Die Fuehrung durch das Programm erfolgt menuegesteuert. Im folgenden sollen Sie noch einige Erlaeuterungen dazu erhalten.

Bildschirm - Anpassung

Eine Anpassung an Ihren speziellen Bildschirm macht sich aufgrund der Tatsache erforderlich, dass verschiedene REDABAS-Befehle (z.B. EDIT, APPEND, BROWSE, MODIFY) den gesamten Bildschirm ausnutzen (im folgenden wird diese Betriebsart in der Terminologie von REDABAS "Seitenmodus" genannt). Mit speziellen Funktionstasten koennen Sie im Seitenmodus den sogenannten Cursor frei bewegen und zeichenweise, feldweise oder satzweise positionieren. Da verschiedene Bildschirmtypen intern unterschiedliche Steuerzeichen fuer die Cursorpositionierung verwenden, ist die Bildschirmanpassung fuer die obengenannten REDABAS-Befehle wichtig.

Standardmaessig ist REDABAS im SCP auf den Bildschirm BAB2 (24 Zeilen * 80 Spalten) und im SCP1700 auf den Bildschirm des A7100 (25 Zeilen * 80 Spalten) angepasst. Wenn Sie einen dieser Bildschirme verwenden, ist keine Neuinstallierung erforderlich.

Haben Sie sich fuer eine Neuinstallierung entschieden, bietet Ihnen das Installierungsprogramm im ersten Auswahlmenue eine Uebersicht der von REDABAS unterstuetzten Bildschirmtypen (bzw. Computer) an. Durch Eintippen eines Buchstaben A ... P koennen Sie einen bestimmten Bildschirmtyp auswaehlen.

Bei der Antwort "Q" haben Sie die Moeglichkeit, die Steuerzeichen fuer einzelne Bildschirmfunktionen zu veraendern. Anhand des Folgemenues koennen Sie diese Funktionen gezielt auswaehlen.

Mit der Antwort "R" bekunden Sie den Wunsch, saemtliche von REDABAS verwendeten Bildschirmsteuerzeichen neu zu definieren.

Die Modifizierung und Neudefinition setzt voraus, dass Sie die erforderlichen Steuerzeichen kennen, gewoehnlich sind diese in der Hardwarebeschreibung erklaert

Modifizieren von Programmparametern

Im zweiten Teil der Installierung koennen Sie folgende REDABAS-Optionen aendern:

- Benennung eines Zeichens als Makroanzeiger (Standard = "&")
- Fehlerkorrekturdialog ein- oder ausschalten (Standard = "aus")
- Datumformat auswaehlen (Standard = Tag.Monat.Jahr)

Arbeiten Sie erstmalig mit REDABAS empfehlen wir Ihnen, zunaechst die Standardgenerierung beizubehalten.

Erst spaeter, wenn Sie z.B. die Bedeutung des Makroanzeigers kennengelernt haben oder Sie das Datum in der Form "Monat.Tag.Jahr" bevorzugen, ist eine Neuinstallierung sinnvoll.

Mit dem Fehlerkorrekturdialog haben Sie die Moeglichkeit, Eingabefehler innerhalb von Befehlszeilen, die REDABAS als Syntaxfehler erkennt, zeichenweise zu korrigieren. Bei langen Befehlszeilen ist diese Variante vorteilhaft.

Durchfuehren der Installierung

Die Installierung wird durch Eintippen des Programmnamens

RDBASINS

aufgerufen. Beachten Sie bitte dabei, dass sich dazu das Programm REDABAS.COM (SCP) bzw. REDABAS.CMD (SCP1700) auf einer Diskette im aktuellen Laufwerk befinden muss, andernfalls erscheint eine Fehlernachricht.

Bei Programmabbruch infolge einer falschen Parametereingabe bleibt die urspruengliche Installierung erhalten.

Erst, wenn Sie am Ende der Installierung die Frage

INSTALLIERUNG AUSFUEHREN ("J") ODER PROGRAMM ABBRECHEN ("N")?

mit "J" beantworten, wird das Ergebnis in REDABAS.COM bzw. REDABAS.CMD gespeichert. Sie koennen also durchaus die Installierung probeweise ausfuehren und am Ende das Ergebnis wieder verwerfen.

2.3. Allgemeine Hinweise

Bevor Sie mit der eigentlichen Handhabung von REDABAS beginnen, moechten wir Sie mit einigen allgemeinen Hinweisen und den in dieser Programmtechnischen Beschreibung verwendeten Regeln vertraut machen.

- Grossgeschriebene Ausschriften in den Bildschirmbeispielen weisen auf eine Antwort des Systems REDABAS oder einen REDABAS-Befehl.

- Variable Angaben in REDABAS-Befehlen (z.B. Datei- und Feldnamen) sind klein geschrieben.

- "..." wird verwendet, um bestimmte Eingaben (Befehle, Mitteilungen) hervorzuheben. Diese Anfuehrungszeichen duerfen Sie nicht eingeben!

- <RETURN> bedeutet, dass Sie die <RETURN>-Taste, (die auf

mancher Rechnertastatur auch als ET, ET1, ENT oder ENTER bezeichnet ist) druecken sollen. Geben Sie nicht das Wort RETURN und auch nicht die Klammern ein!
Das Gleiche gilt fuer die <CONTROL>-Taste!

- Fuer die Notation von REDABAS-Befehlen gilt:

  [...] Schrift in eckigen Klammern ist Teil eines REDABAS-Befehls, der wahlweise angegeben werden kann.

  <...> Kleingeschriebenes in spitzen Klammern ist Teil eines REDABAS-Befehls, der mit echter Information ausgefuellt werden muss.
  Beispiel: CREATE <dateiname>
  Sie muessen anstelle "dateiname" den Namen einer zu erzeugenden Datei eingeben, z.B. "kunden".

- Grundregeln der REDABAS-Befehlssprache:

Es ist im allgemeinen gleichgueltig, ob Befehle in Gross- oder Kleinbuchstaben geschrieben werden. Wegen der besseren Unterscheidung und Lesbarkeit von Programmen wird jedoch empfohlen, Befehle gross und Datei- und Feldnamen klein zu schreiben.

Ein Befehl darf max. 254 Zeichen lang sein. Wird auf dem Bildschirm die Laenge einer Zeile ueberschritten, ist der Teil des Befehls mit Semikolon abzuschliessen, die <RETURN>-Taste zu druekken und der Befehl auf der naechsten Zeile fortzusetzen.

In einer Befehlszeile muss das erste, vom Leerzeichen verschiedene Zeichen zum Befehlswort selbst gehoeren. Die Zusatzklauseln koennen in beliebiger Reihenfolge angehaengt werden. Ein Befehlswort ist z.B. "CREATE" oder "INSERT". Eine Zusatzklausel spezifiziert die durch ein Befehlswort eingeleitete Aktion, z.B. wird mit "FOR" eine Zusatzklausel eingeleitet.

Die Gesamtheit von Befehlswoertern und Einleitungswoertern von Zusatzklauseln bilden die sogenannten "Steuerwoerter" von REDABAS. Sie sind in der Programmtechnischen Beschreibung immer in Grossbuchstaben geschrieben.

Steuerwoerter werden durch ihre ersten vier Buchstaben eindeutig identifiziert, d.h. Steuerwoerter duerfen auf vier Buchstaben abgekuerzt werden. Beispielsweise kann DISPLAY STRUCTURE auf DISP STRU verkuerzt werden.

Nach der Eingabe eines Befehles ist die <RETURN>-Taste zu betaetigen.

Damit Sie REDABAS schon nach kurzer Einarbeitungszeit nutzen koennen, werden in den Abschnitten 3. und 4. dieser Programmtechnischen Beschreibung die REDABAS-Befehle und

-Funktionen nach Moeglichkeit in der Reihenfolge und Notation vorgestellt (und an Beispielen demonstriert), in der Sie sie fuer die praktische Arbeit brauchen. Bewusst wurde dabei auf das vollstaendige Befehlsspektrum und auch auf die vollstaendige Beschreibung aller Moeglichkeiten eines Befehls verzichtet. Im Abschnitt 8. finden Sie eine lueckenlose Darstellung saemtlicher Befehle mit all ihren Varianten in alphabetischer Reihenfolge geordnet, im Abschnitt 7.4. finden Sie alle REDABAS-Funktionen dargestellt.

2.4. Starten und Beenden

Wenn Sie mit REDABAS arbeiten wollen, fuehren Sie bitte folgende Handlungen aus:

- Legen sie eine Diskette mit dem Betriebssystem SCP ins Laufwerk A und schalten Sie den Computer ein!

- Das Betriebssystem meldet seine Dialogbereitschaft durch die Anzeige "A >" auf dem Bildschirm.

- Gewoehnlich wird sich auf der Diskette mit den REDABAS-Programmkomponenten auch das Betriebssystem befinden und Sie koennen sofort durch Eintippen von "REDABAS" und Betaetigen der <RETURN>-Taste den Dialog mit dem Datenbankbetriebssystem beginnen.

- Haben Sie REDABAS getrennt vom Betriebssystem auf einer anderen Diskette gespeichert, so ist das Laufwerk, in das Sie die REDABAS-Programmdiskette einlegen, zunaechst als aktuelles Laufwerk zuzuweisen (durch Eingabe von B: oder C: <RETURN>), bevor Sie das Datenbanksystem durch Eintippen von "REDABAS" und <RETURN> starten.
  Wenn Sie REDABAS nicht vom aktuellen Laufwerk aufrufen, tritt ein Ladefehler auf!
  Unter SCP1700 besteht die Moeglichkeit, einen Teil des Arbeitsspeichers wie ein Diskettenlaufwerk zu benutzen (memory-disk E:). Wenn sie die REDABAS-Programmkomponenten vor dem Start auf dieses Laufwerk (E:) kopieren, erreichen Sie damit eine wesentliche Beschleunigung bei der Programmabarbeitung

- Nach erfolgreichem Start meldet sich REDABAS mit der Ausschrift

```
   ROBOTRON-PROJEKT DRESDEN

   *****************
   *    REDABAS    *
   *****************
   Version 2.0 - 17.Feb.1987        (unter SCP)
bzw.
   Version 5.0 - 26.Mai 1987        (unter SCP1700)
```

  und gibt Ihnen anschliessend die Moeglichkeit, das aktuelle

Datum in der Form

TT.MM.JJ oder TT/MM/JJ (Tag/Monat/Jahr)

einzugeben.
Die Eingabe des Datums ist wahlweise. Haben Sie sich fuer die Eingabe des Datums zum jetzigen Zeitpunkt entschieden, speichert REDABAS dies fuer spaetere Abrufe und fuegt es bei Listen in die Kopfinformationen ein.

- Nun fordert Sie REDABAS zur Eingabe eines Kommandos auf und zeigt die Eingabebereitschaft durch einen Doppelpunkt ":"an. Ab hier beginnt grundsaetzlich die Arbeit mit REDABAS. Sie koennen eine Anweisung, eine Frage oder einen Aufruf zur Abarbeitung eines Programms eingeben.

- Will man das Programm REDABAS beenden, geschieht das immer mit dem Befehl QUIT. Nach der Eingabe von QUIT erscheint die REDABAS-Ende-Nachricht:

```
******************
* REDABAS - ENDE *
******************
```

Anschliessend meldet sich wieder das Betriebssystem.

Noch eine Bemerkung zur Arbeit mit mehreren Diskettenlaufwerken. Wenn es Ihre Geraetekonfiguration zulaesst, ist es ratsam, die REDABAS-Datenbank nicht auf der REDABAS-Programmdiskette aufzubauen.

Sie koennen alle vom Betriebssystem zugelassenen Laufwerke fuer die REDABAS-Dateien nutzen, wenn Sie vor den Dateinamen die Bezeichnung des Laufwerkes mit einem Doppelpunkt setzen.

Moechten Sie beispielsweise Laufwerk B benutzen, geben Sie an: B:kunden. Eine Datei kann sich maximal ueber eine Diskette erstrecken. Es ist nicht notwendig, dass eine Datenbankdatei und andere zur Datenbank gehoerende Dateien auf ein- und derselben Diskette angeordnet werden.

3. Funktionen der Datenbankverwaltung

3.1. Einrichten und Aendern von Dateien

3.1.1. Erstellen einer Datei

Eine effektive Datenbankverwaltung erfordert Kenntnis des Informationsbedarfs der einzelnen Sachgebiete. Entsprechend erfolgt der logische Datenbankentwurf, d.h. konkret das Festlegen der Struktur der Dateien.

Beim Erstellen einer Datei wird also gewissermassen der aeussere Rahmen festgelegt, in den die Daten aufgenommen werden sollen. Geht man von der Sicht des Anwenders aus, so sieht er seine als Datei abzuspeichernden Datenbestaende in Form einer Tabelle, deren Zeilen Datensaetzen und deren Spalten Datenfeldern entsprechen. Im Beispiel der Kundendatei ist eine komplette Information ueber den Kunden (name, vorname usw.), also eine Zeile, als Datensatz und die Menge der gleichartigen Datenelemente (z.B. alle Namen), also eine Spalte, als Datenfeld realisiert.

Alle Datensaetze einer Datei (Zeilen) sind gleich strukturiert, d.h. dass nur die Struktur eines Datensatzes beschrieben werden braucht.

Im folgenden moechten wir Ihnen anhand eines Beispiels die mit REDABAS ausfuehrbaren Funktionen der Datenbankverwaltung - beginnend mit dem Erstellen einer Datei - demonstrieren.

- Ist REDABAS bereit? (Es muss ein Doppelpunkt auf dem Bildschirm erscheinen.)

Eine Datei wird mit dem Befehl

```
CREATE <dateiname>
```

erstellt.
Der Dateiname darf bis zu 8 Zeichen lang sein und muss mit einem Buchstaben beginnen. Als Demonstrationsbeispiel erstellen wir eine Datei namens "kunden".
Geben Sie bitte ein:

```
CREATE kunden
```

REDABAS erzeugt eine Datei mit dem Namen KUNDEN.DBD. (Der Teil des Dateinamens nach dem Punkt ist der SCP-Dateityp und wird von REDABAS automatisch vergeben.)
REDABAS ruft eine Routine zur Strukturierung der Datensaetze auf. Auf dem Bildschirm erscheint folgende Ausschrift:

```
GEBEN SIE DIE SATZSTRUKTUR WIE FOLGT EIN:
 FELD    NAME,TYP,LAENGE,ANZAHL DEZIMALSTELLEN
  001
```

Sie geben jetzt zeilenweise die Angaben zu den einzelnen Datenfeldern ein. Die Felder sind numeriert. maximal sind 32 Felder moeglich. Folgende Angaben sind erforderlich:

NAME Bezeichnung des Datenfeldes
max. 10 Zeichen, beginnend mit einem Buchstaben, danach sind auch Ziffern und Doppelpunkt erlaubt, jedoch keine Leerstellen und keine anderen Sonderzeichen (einschliesslich Umlaute und "sz").

TYP Der Typ des Datenfeldes wird durch einen einzelnen Buchstaben spezifiziert:

C: Alphanumerisch (am meisten verwendet)
Alle druckbaren ASCII-Zeichen, einschliesslich der Ziffern, Sonderzeichen und des Leerzeichens

N: Numerisch
Positive und negative Zahlen von $1 * 10^{63}$ bis herab zu $1.0 * 10^{-63}$. Die Genauigkeit betraegt 10 Stellen oder bis auf den Pfennig genau bei Betraegen, die nicht groesser als 99 999 999.99 M sind.

L: Logisch
Darunter versteht man die Werte "wahr" und "falsch" oder "ja" und "nein". REDABAS erkennt J,j und T,t als "wahr" und N,n und F,f als "falsch".

LAENGE Feldlaenge
Bis zu 254 Zeichen kann die Feldlaenge sein. Bei numerischen Feldern muss eine Stelle fuer den Dezimalpunkt beruecksichtigt werden und eine weitere Stelle, wenn Vorzeichen gewuenscht werden.

DEZIMAL-STELLEN Stellenzahl nach dem Dezimalpunkt, wenn Dezimalstellen gewuenscht werden

Die einzelnen Parameter sind untereinander mit Komma zu trennen. Sollten Ihnen Fehler bei der Eingabe unterlaufen, bitte beachten Sie:

- Innerhalb der Zeile, die Sie gerade eingeben (auf der sich der Cursor befindet), koennen Sie sofort durch Ruecksetzen des Cursors die falsche Eingabe ueberschreiben.
  Haben Sie die Zeile bereits verlassen, gibt es zwei Moeglichkeiten:

- REDABAS hat den Fehler erkannt, das betreffende Feld wird erneut zur Eingabe aufgerufen.

- Erkennt REDABAS den Fehler nicht, vervollstaendigen Sie zunaechst die Eingabe der Satzstruktur und korrigieren Sie den Fehler spaeter. Dazu dient der spaeter beschriebene Befehl MODIFY STRUCTURE.

Definieren Sie jetzt die Datei "kunden" wie folgt:

```
-------------------------------------------------
| GEBEN SIE DIE SATZSTRUKTUR WIE FOLGT EIN:     |
|  FELD    NAME,TYP,LAENGE,ANZAHL DEZIMALSTELLEN |
|  001     name,c,10                            |
|  002     vorname,c,8                          |
|  003     plz,c,4                              |
|  004     ort,c,8                              |
|  005     strasse,c,15                         |
|  006                                          |
|                                               |
-------------------------------------------------
```

Bild 1 Struktur der Datei "kunden"

Beachten Sie, dass beispielsweise das Feld plz (Postleitzahl) als alphanumerisch definiert wurde, obwohl es normalerweise aus Ziffern besteht. Da keinerlei arithmetische Operationen damit ausgefuehrt werden, ist dies moeglich.

Die Definition der Struktur wird durch Druecken der <RETURN>-Taste abgeschlossen. REDABAS beendet und speichert die Struktur. Am Bildschirm erscheint die Frage:

MOECHTEN SIE JETZT DATEN EINGEBEN? (J/N)

Haben Sie zuerst Eingabefehler in der Struktur zu korrigieren, so geben Sie "N" (Nein) und anschliessend "MODIFY STRUCTURE" (s. Abschnitt 3.3.) ein.

Beantworten Sie die Frage mit "J" (Ja), wird der Bildschirm geloescht. Sie beginnen jetzt mit der Eingabe Ihrer Daten. REDABAS zeigt Ihnen eine Maske zur Datenerfassung entsprechend der definierten Struktur:

```
-----------------------------------
| SATZ# 00001                     |
| NAME      :           :         |
| VORNAME   :         :           |
| PLZ       :    :                |
| ORT       :         :           |
| STRASSE   :               :     |
-----------------------------------
```

Bild 2 Maske fuer die Datenerfassung

Von REDABAS wird automatisch eine fortlaufende Satznummer vergeben, hier beginnend mit 00001 (Anfang der Datei). Pro Datei koennen maximal 65535 Datensaetze gespeichert werden.

Entsprechend Ihrer vorhin definierten Datenfeldlaenge erscheint am Ende jeder Zeile ein Doppelpunkt. Sobald Sie das Feld vollstaendig ausgefuellt oder <RETURN> gedrueckt haben, rueckt der Cursor auf das naechste Feld. Fuer Korrekturen koennen Sie den Cursor mittels der Cursortasten oder den Tastenkombinationen <CONTROL-E>, <CONTROL-A>, <CONTROL-X>, <CONTROL-F>, <CONTROL-S> und <CONTROL-D> (vergleiche die Tabelle der bildschirm-orientierten Tastaturbefehle in diesem

Abschnitt und siehe Abschnitt 6.2.) innerhalb der Maske vor- und zurueckbewegen. Bitte achten Sie bei der Dateneingabe auf Gross- und Kleinschreibung, da bei spaeteren Auswertungen der Datenbank auch dies fuer Unterscheidungen herangezogen wird.

Sind Ihnen bei der Eingabe numerischer Feldinhalte formale Fehler unterlaufen, zeigt Ihnen das REDABAS durch "***" im betreffenden Feld auf dem Bildschirm an und speichert diesen Wert nicht in der Datei.

Nachdem das letzte Feld (im Beispiel STRASSE) in einem Datensatz ausgefuellt wurde, wird die <RETURN>-Taste betaetigt und es kann mit der Eingabe des naechsten Datensatzes begonnen werden.

Geben Sie bitte die folgenden Datensaetze der Datei "kunden" ein:

```
Lehmann     Kurt    8060 DRESDEN    Landstr. 15
Schmidt     Gisela  8223 THARANDT   Bahnhofstr. 43
Moser       Otto    8212 FREITAL    Neugraben 8
Weber       Franz   8223 THARANDT   Hortgasse 14
Schulze     Ernst   8060 DRESDEN    Nebenstr. 61
Miller      Peter   8212 FREITAL    Nelkenweg 23
Meier       August  8122 RADEBEUL   Uferstr. 12
Neumann     Inge    8222 RABENAU    Gartenstr. 4
Fleischer   Karl    8212 FREITAL    Am See 83
Schmidt     Heinz   8223 THARANDT   Waldweg 42
```

Sie haben jetzt 10 Datensaetze eingegeben. Sie beenden die Erfassung durch Druecken der <RETURN>-Taste, wenn die Maske fuer Satz Nr.11 angezeigt wird.

Mit der Eingabe dieser Saetze hat REDABAS sie bereits auf der Diskette gespeichert. Wollen Sie jetzt Ihre Arbeit beenden, geben Sie den Befehl QUIT ein.

Moechten Sie weiterarbeiten, ist es zweckmaessig, eine Liste Ihrer bisherigen Eingaben anzufertigen. Sie benoetigen dazu die Befehle

USE <dateiname> und
LIST

Mit dem Befehl USE wird eine bereits vorhandene Datenbankdatei solange aktiviert, bis ein neuer USE-Befehl eine andere Datei aufruft.Unsere Datei "kunden" muss also mit USE eroeffnet werden. Anschliessend soll der Inhalt dieser Datei aufgelistet werden.

Sie geben bitte ein:

USE kunden <RETURN>
LIST <RETURN>

Das Druecken der <RETURN>-Taste muss nach jedem Befehl erfolgen. Im folgenden wird dies deshalb nicht mehr ausdruecklich erwaehnt.

Wir erhalten eine Liste aller Datensaetze, versehen mit einer von REDABAS vergebenen laufenden Nummer.

```
:USE kunden
:LIST
00001  Lehmann     Kurt    8060 DRESDEN    Landstr. 15
00002  Schmidt     Gisela  8223 THARANDT   Bahnhofstr. 43
00003  Moser       Otto    8212 FREITAL    Neugraben 8
00004  Weber       Franz   8223 THARANDT   Hortgasse 14
00005  Schulze     Ernst   8060 DRESDEN    Nebenstr. 61
00006  Miller      Peter   8212 FREITAL    Nelkenweg 23
00007  Meier       August  8122 RADEBEUL   Uferstr. 12
00008  Neumann     Inge    8222 RABENAU    Gartenstr. 4
00009  Fleischer   Karl    8212 FREITAL    Am See 83
00010  Schmidt     Heinz   8223 THARANDT   Waldweg 42
```

Bild 3 Liste der Datei "kunden"

Wollen Sie die Liste auf dem Drucker ausgeben, betaetigen Sie die <CONTROL-P>-Tastenkombination und wiederholen Sie den Befehl LIST Das Abschalten des Druckers erfolgt durch nochmaliges Druecken von <CONTROL-P>.

Mit LIST koennen Sie sich auch die Struktur Ihrer Datei anzeigen lassen, Sie erweitern den Befehl auf LIST STRUCTURE.

```
: LIST STRUCTURE

DATEISTRUKTUR FUER: KUNDEN  .DBD
ANZAHL DER SAETZE:  00010
DATUM DER LETZTEN AENDERUNG: 12.02.85
DATENBANKDATEI IM PRIMAERZUGRIFF
FELD     NAME       TYP  LAENGE  DEZ
001      NAME        C    010
002      VORNAME     C    008
003      PLZ         C    004
004      ORT         C    008
005      STRASSE     C    015
** SUMME **           00046
```

Bild 4 Struktur der Datei "kunden"

Es bedeuten:

DATEISTRUKTUR FUER : KUNDEN .DBD
Der Dateityp "DBD" weist darauf hin, dass es sich um eine Datenbank-Datei handelt.

DATUM DER LETZTEN AENDERUNG: 12.02.85
Das Datum einer Aenderung kann das Aktualisieren sowohl der Daten als auch der Dateistruktur betreffen.

** SUMME ** bezieht sich auf die Gesamtlaenge eines Datensatzes. Es wurden die einzelnen Feldlaengen addiert und

ein Zeichen hinzugefuegt, dessen Bedeutung spaeter unter der Ueberschrift "Loeschen von Saetzen" erklaert wird.

Es ist unbedingt zu empfehlen, die Dateistruktur auszudrucken, da Sie zu jeder Zeit ueber Namen und Typ der Datenfelder wegen spaeter auszufuehrender Operationen Bescheid wissen muessen.

3.1.2. Eingeben weiterer Datensaetze

Fuer das An- bzw. Einfuegen weiterer Datensaetze stehen die Befehle APPEND und INSERT zur Verfuegung.

APPEND

ermoeglicht das Anfuegen von Saetzen an eine vorhandene Datei. Haben Sie APPEND eingegeben, antwortet REDABAS mit dem Anzeigen der Eingabemaske.

Im Beispiel moechten wir die Datei "kunden" um drei Kunden erweitern.

Wir geben ein:

```
USE kunden
APPEND
```

Es erscheint die Eingabemaske mit Satz Nr.11:

```
------------------------------------
|                                   |
| SATZ# 00011                       |
| NAME      :           :           |
| VORNAME   :         :             |
| PLZ       :     :                 |
| ORT       :         :             |
| STRASSE   :                 :     |
|                                   |
------------------------------------
```

Bild 5 Maske fuer das Anfuegen eines neuen Datensatzes

Geben Sie die folgenden 3 Datensaetze ein:

```
Lolle      Peter    8122 RADEBEUL  Goethestr. 13
Heinze     Max      8060 DRESDEN   Rosenstr. 78
Tauber     Fred     8028 DRESDEN   Fliederstr. 42
```

Die Datei "kunden" enthaelt damit 13 Saetze. Wir listen nun die vollstaendige Datei auf:

```
: USE kunden
: LIST

00001  Lehmann   Kurt    8060 DRESDEN    Landstr. 15
00002  Schmidt   Gisela  8223 THARANDT   Bahnhofstr. 43
00003  Moser     Otto    8212 FREITAL    Neugraben 8
00004  Weber     Franz   8223 THARANDT   Hortgasse 14
00005  Schulze   Ernst   8060 DRESDEN    Nebenstr. 61
00006  Miller    Peter   8212 FREITAL    Nelkenweg 23
00007  Meier     August  8122 RADEBEUL   Uferstr. 12
00008  Neumann   Inge    8222 RABENAU    Gartenstr. 4
00009  Fleischer Karl    8212 FREITEAL   Am See 83
00010  Schmidt   Heinz   8223 THARANDT   Waldweg 42
00011  Lolle     Peter   8122 RADEBEUL   Goethestr. 13
00012  Heinze    Max     8060 DRESDEN    Rosenstr. 78
00013  Tauber    Fred    8028 DRESDEN    Fliederstr. 42
```

Bild 6 Erweiterte Datei "kunden"

INSERT

ermoeglicht das Einfuegen von Saetzen an jeder beliebigen Stelle der Datei . Fuer das vorausgehende Positionieren auf einen bestimmten Datensatz gibt es verschiedene Moeglichkeiten. Das Einfachste ist es, anstelle eines Befehls die Datensatznummer einzugeben (z.B. : 4). Der einzufuegende Satz kann vor oder nach dem aktuellen Satz eingeordnet werden.

3.1.3. Aendern von Datensaetzen

Ergeben sich Veraenderungen an bereits erfassten und abgespeicherten Datensaetzen, bietet REDABAS drei Moeglichkeiten der Aenderung. Die Befehle lauten:

EDIT
CHANGE
REPLACE

Zunaechst moechten wir Ihnen anhand des Demonstrations-beispiels die Wirkungsweise von EDIT vorfuehren.

Es ist die Anschrift des Kunden Franz Weber, Datensatz Nr. 4 , zu veraendern.

Eingabe:

```
USE kunden
EDIT
```

Darauf erscheint die Systemnachricht:

GEBEN SIE DIE DATENSATZ-NR. EIN:

Sie geben ein:

4

Auf dem Bildschirm erscheint die Maske mit den Daten des Satzes 4:

```
-------------------------------------------
|                                         |
|  SATZ # 00004                           |
|  NAME       :Weber      :               |
|  VORNAME    :Franz    :                 |
|  PLZ        :8223:                      |
|  ORT        :THARANDT:                  |
|  STRASSE    :Hortgasse 14    :          |
|                                         |
-------------------------------------------
```

Bild 7 (Unveraenderter) Datensatz Nr. 00004

Positionieren Sie den Cursor (mit CONTROL-X) auf das Datenfeld STRASSE und aendern dieses wie folgt:

```
-------------------------------------------
|                                         |
|  SATZ # 00004                           |
|  NAME       :Weber      :               |
|  VORNAME    :Franz    :                 |
|  PLZ        :8223:                      |
|  ORT        :THARANDT:                  |
|  STRASSE    :Heideweg 66     :          |
|                                         |
-------------------------------------------
```

Bild 8 Veraenderter Datensatz Nr.00004

Mit <CONTROL-W> wird die Bearbeitung des Datensatzes abgeschlossen. Der veraenderte Datensatz wird damit gespeichert und es erscheint wiederum die Systemnachricht:

GEBEN SIE DIE DATENSATZ-NR. EIN:

Wollen Sie keine weiteren Datensaetze bearbeiten, schliessen Sie mit <RETURN> ab.

Wenn Sie nach Aenderung des letzten Datenfeldes <RETURN> eingeben, zeigt REDABAS sofort den naechsten Datensatz an.

Haben Sie nur einen einziger Datensatz zu aendern, ist es zweckmaessig, die Datensatznummer direkt im EDIT-Befehl zu nennen, beispielsweise

EDIT 4

Nach Bearbeitung dieses Satzes verlaesst REDABAS die EDIT-Funktion sofort und fragt nicht nach einer weiteren Satznummer.

Durch LIST lassen wir uns die Datei "kunden" nach der Aenderung nochmals anzeigen:

```
: USE kunden
: LIST

00001  Lehmann   Kurt    8060 DRESDEN  Landstr. 15
00002  Schmidt   Gisela  8223 THARANDT Bahnhofstr. 43
00003  Moser     Otto    8212 FREITAL  Neugraben 8
00004  Weber     Franz   8223 THARANDT Heideweg 66
00005  Schulze   Ernst   8060 DRESDEN  Nebenstr. 61
00006  Miller    Peter   8212 FREITAL  Nelkenweg 23
00007  Meier     August  8122 RADEBEUL Uferstr. 12
00008  Neumann   Inge    8222 RABENAU  Gartenstr. 4
00009  Fleischer Karl    8212 FREITAL  Am See 83
00010  Schmidt   Heinz   8223 THARANDT Waldweg 42
00011  Lolle     Peter   8122 RADEBEUL Goethestr. 13
00012  Heinze    Max     8060 DRESDEN  Rosenstr. 78
00013  Tauber    Fred    8028 DRESDEN  Fliederstr. 42
```

Bild 9 Datei "kunden" nach der Aenderung

Im folgenden geben wir Ihnen einen Ueberblick ueber die innerhalb der Funktion EDIT moeglichen Tastenkombinationen und deren Funktionen. (Die Kombination besteht immer aus der <CONTROL>-Taste und einer Buchstabentaste, wobei die Buchstabentaste bei niedergedrueckter <CONTROL>-Taste zu betaetigen ist.)

Bildschirmorientierte Tastaturbefehle im "EDIT"-Modus:

CONTROL-W Schreiben des aktuellen Datensatzes auf die Diskette und Beenden der Funktion EDIT

CONTROL-C Schreiben des (geaenderten) Datensatzes auf die Diskette und Aufrufen des naechsten Satzes zur Bearbeitung

CONTROL-R Schreiben des (geaenderten) Satzes auf die Diskette und Aufrufen des vorhergehenden Satzes zur Bearbeitung

CONTROL-Y Loeschen des Teils des laufenden Feldes rechts vom Cursor durch Auffuellen mit Leerstellen

CONTROL-U (Ver.1.0) CONTROL-L (Ver.2.0) Ein-/ Ausschalten der Loeschmarkierung fuer einen Datensatz. Beim Einschalten erscheint am oberen Rand der Satzmaske das Wort "GELOESCHT". Drueckt man <CONTROL-U> nochmals, kann die Loeschung rueckgaengig gemacht werden.

CONTROL-V Umschalten zwischen Ueberschreiben und Einfuegen von Zeichen
Wird "Einfuegen" gewaehlt, wird das auf dem Bildschirm rechts oben angezeigt.

CONTROL-Q Abbruch der Aenderungen im gerade bearbeiteten Datensatz und Anfordern einer neuen Datensatznummer. Aenderungen im aktuellen Datensatz bleiben unwirksam.

Allgemeine bildschirmorientierte Tastaturbefehle

CONTROL-G Loeschen des durch den Cursor positionierten Zeichens

CONTROL-X CONTROL-F Bewegen des Cursors nach unten zum naechsten Feld

CONTROL-E CONTROL-A Bewegen des Cursors zurueck zum vorhergehenden Feld

CONTROL-D Bewegen des Cursors um ein Zeichen vorwaerts

CONTROL-S Bewegen des Cursors um ein Zeichen zurueck

Eine zweite Moeglichkeit der Veraenderung von Datensaetzen ist die Anwendung des Befehls

CHANGE <bereich> FIELD <feldnamen>

Mit geringem Aufwand koennen Sie den Inhalt einzelner Datenfelder aendern. REDABAS unterscheidet zwischen Feldern alphanumerischen und numerischen Typs. Bei numerisch definierten Feldern (Typ N) wird Ihnen nach der Eingabe des

entsprechend formulierten Befehls die Satz-Nr. und der bisherige Datenfeldinhalt angezeigt. Nach der Ausschrift "AENDERN AUF" geben Sie Ihren neuen Wert ein. Die Aenderung eines alphanumerisch definierten Feldes (Typ C) demonstrieren wir Ihnen wieder an der Kundendatei im Bild 10. Der Kunde "Schulze" ist von der Nebenstrasse auf die Hauptstrasse verzogen. Es ist also das Feld "strasse" zu veraendern. Die letzte Frage ist mit <RETURN> zu beantworten.

```
------------------------------------------------------
|                                                    |
| : USE kunden                                       |
| : LIST FOR name = 'Schulze'                        |
| 00005 Schulze   Ernst   8060 DRESDEN  Nebenstr. 61 |
|                                                    |
| : CHANGE RECORD 5 FIELD strasse                    |
|                                                    |
| SATZ-NR. 00005                                     |
|                                                    |
| STRASSE: Nebenstr. 61                              |
| AENDERN VON ?Neben                                 |
| AENDERN AUF: Haupt                                 |
|                                                    |
| STRASSE: Hauptstr. 61                              |
| AENDERN VON ?                                      |
|                                                    |
------------------------------------------------------
```

Bild 10 Aendern eines Datensatzes mit CHANGE

Weiterhin koennen Sie einige oder auch alle Datensaetze mit dem Befehl

REPLACE <feldname> WITH <ausdruck>

aendern. Dieser Befehl ist sehr maechtig und kann folgendermassen interpretiert werden:
Ersetze den Inhalt des Feldes <feldname> durch den Inhalt oder das Ergebnis von <ausdruck>.
<ausdruck> kann ein anderer Feldname, ein Text, eine Zahl oder eine Formel sein.
Will man das "Ersetzen" an allen Saetzen einer Datei vornehmen, wird der Befehl erweitert:

REPLACE ALL <feldname> WITH <ausdruck>

Diese grosszuegige Moeglichkeit der Manipulierung einer ganzen Datei bietet nur ein relationales Datenbankbetriebssystem. Sie werden bei der weiteren Arbeit mit dieser Programmtechnischen Beschreibung noch die Anwendung des Befehls REPLACE kennenlernen

3.1.4. Loeschen von Datensaetzen

Sollen Datensaetze aus der Datei geloescht werden, kann dies in zwei Varianten erfolgen.

Die erste Moeglichkeit besteht in der Anzeige des zu loeschenden Satzes mit dem EDIT-Befehl und dem Eintragen der

Markierung zum Loeschen durch <CONTROL-L>. Am oberen Bildschirmrand erscheint der Hinweis "GELOESCHT".

Die zweite Moeglichkeit der Loeschung von Datensaetzen bietet der Befehl

DELETE

Mit diesem Befehl koennen Sie einen Datensatz unter Angabe der Satznummer mit dem Loeschkennzeichen versehen. Ausserdem ist es moeglich, einen Datensatzbereich oder auch alle Datensaetze anzugeben oder Auswahlbedingungen zu spezifizieren.

Nach beiden Loeschvarianten bleiben die Datensaetze physisch erhalten. Sie werden in einer dafuer reservierten Stelle im Datensatz (von der schon vorn gesprochen wurde) mit der Markierung "*" versehen.

Wir moechten Ihnen anhand unseres Beispiels das Loeschen von Datensaetzen demonstrieren.
Im Bild 11 sehen Sie die Datei "kunden" aufgelistet, wobei die Datensaetze Nr. 6, 8 und 11 durch folgende Befehle markiert worden sind

```
USE kunden
DELETE RECORD 6
DELETE RECORD 8
DELETE RECORD 11
```

```
: LIST

00001  Lehmann   Kurt     8060 DRESDEN   Landstr. 15
00002  Schmidt   Gisela   8223 THARANDT  Bahnhofstr. 43
00003  Moser     Otto     8212 FREITAL   Neugraben 8
00004  Weber     Franz    8223 THARANDT  Heideweg 66
00005  Schulze   Ernst    8060 DRESDEN   Hauptstr. 61
00006 *Miller    Peter    8212 FREITAL   Nelkenweg 23
00007  Meier     August   8122 RADEBEUL  Uferstr. 12
00008 *Neumann   Inge     8222 RABENAU   Gartenstr. 4
00009  Fleischer Karl     8212 FREITAL   Am See 83
00010  Schmidt   Heinz    8223 THARANDT  Waldweg 42
00011 *Lolle     Peter    8122 RADEBEUL  Goethestr. 13
00012  Heinze    Max      8060 DRESDEN   Rosenstr. 78
00013  Tauber    Fred     8028 DRESDEN   Fliederstr. 42
```

Bild 11 Datei "kunden" mit den zur Loeschung markierten Datensaetzen

Alle Saetze sind jetzt noch vorhanden, die Loeschung koennte jederzeit ohne weiteres rueckgaengig gemacht werden. Die physische Loeschung der Saetze erfolgt erst nach Eingeben des Befehls PACK.(Fuehren Sie aber bitte diesen Befehl jetzt nicht aus!)

Bild 12 zeigt Ihnen die Datei "kunden" , nachdem die Loeschung erfolgt ist. Die verbliebenen Datensaetze sind mit neuen fortlaufenden Nummern abgespeichert.

```
------------------------------------------------------------
|                                                          |
| : PACK                                                   |
|                                                          |
| DATENBANKDATEI VERDICHTET, 00010 SAETZE KOPIERT          |
|                                                          |
| : LIST                                                   |
|                                                          |
| 00001  Lehmann   Kurt     8060 DRESDEN   Landstr. 15     |
| 00002  Schmidt   Gisela   8223 THARANDT  Bahnhofstr. 43  |
| 00003  Moser     Otto     8212 FREITAL   Neugraben 8     |
| 00004  Weber     Franz    8223 THARANDT  Heideweg 66     |
| 00005  Schulze   Ernst    8060 DRESDEN   Hauptstr. 61    |
| 00006  Meier     August   8122 RADEBEUL  Uferstr. 12     |
| 00007  Fleischer Karl     8212 FREITAL   Am See 83       |
| 00008  Schmidt   Heinz    8223 THARANDT  Waldweg 42      |
| 00009  Heinze    Max      8060 DRESDEN   Rosenstr. 78    |
| 00010  Tauber    Fred     8028 DRESDEN   Fliederstr. 42  |
|                                                          |
------------------------------------------------------------
```

Bild 12 Drei markierte Saetze der Datei "kunden" sind physisch geloescht

3.2. Wiederauffinden aus der Datenbank

Bei der praktischen Nutzung einer Datenbank kommt es vor allem darauf an, nach der Phase der Einrichtung der Datenbank gezielte Zugriffe fuer die Befriedigung unterschiedlicher Informationsbeduerfnisse vorzunehmen.

Sie lernten bereits den Befehl LIST fuer das Auflisten Ihrer Datei kennen. Dieser Befehl kann, um einige Angaben erweitert, die Datei jedoch auch nach bestimmten Kriterien durchsuchen und/oder nur ausgewaehlte Teile der Datei, spezifizierte Datensaetze oder Felder anzeigen.

3.2.1. Selektion bestimmter Datenfelder

Zunaechst moechten wir von der Datei "kunden" nur NAMEN, VORNAMEN und ORT auszugeben lassen
Bitte geben Sie ein:

```
USE kunden
LIST vorname,name,ort
```

Es werden nun nur die spezifizierten Felder ausgegeben, die nicht angefuehrten Felder werden unterdrueckt. Es ist wichtig, dass Sie die Feldnamen genau so schreiben, wie sie definiert wurden.

```
: USE kunden
: LIST vorname,name,ort
00001  Kurt     Lehmann     DRESDEN
00002  Gisela   Schmidt     THARANDT
00003  Otto     Moser       FREITAL
00004  Franz    Weber       THARANDT
00005  Ernst    Schulze     DRESDEN
00006  Peter    Miller      FREITAL
00007  August   Meier       RADEBEUL
00008  Inge     Neumann     RABENAU
00009  Karl     Fleischer   FREITAL
00010  Heinz    Schmidt     THARANDT
00011  Peter    Lolle       RADEBEUL
00012  Max      Heinze      DRESDEN
00013  Fred     Tauber      DRESDEN
```

Bild 13 Datei "kunden", spezifiziert nach Vorname, Name, Ort

3.2.2. Bedingte Auflistung von Datensaetzen

Will man die Auswerung einer Datei in Abhaengigkeit von der Erfuellung bestimmter Bedingungen vornehmen, geschieht dies mit

LIST FOR <bedingung>

Der einfachste Fall einer Bedingung ist, einen Ausdruck vorzugeben und die Datei nach diesem Ausdruck zu durchsuchen.

Datenfeld als Suchbegriff

Im folgenden Beispiel soll die Kundendatei nach den Kunden durchsucht werden,
1. deren Wohnort "DRESDEN" ist:
2. deren Name "Schmidt" ist:

```
: USE kunden
: LIST FOR ort = 'DRESDEN'

00001  Lehmann   Kurt    8060 DRESDEN  Landstr. 15
00005  Schulze   Ernst   8060 DRESDEN  Hauptstr. 61
00012  Heinze    Max     8060 DRESDEN  Rosenstr. 78
00013  Tauber    Fred    8028 DRESDEN  Fliederstr. 42

: LIST FOR name = 'Schmidt'

00002  Schmidt   Gisela  8223 THARANDT Bahnhofstr. 43
00010  Schmidt   Heinz   8223 THARANDT Waldweg 42
```

Bild 14 Datei "kunden", in zwei Varianten durchsucht

Beachten Sie bei der Formulierung bitte folgende Regeln:

- Der Name des zu untersuchenden Datenfeldes darf gross oder klein geschrieben sein, muss aber bezueglich Zeichenfolge mit der Definition entsprechend der Dateistruktur uebereinstimmen.

- Der Ausdruck, der mit dem Inhalt der Datenfelder verglichen werden soll, muss neben genauer Uebereinstimmung in der Schreibweise auch bezueglich Gross- und Kleinschreibung mit den gespeicherten Daten uebereinstimmen. (Im Beispiel wurden in der Kundendatei die Ortsnamen mit Grossbuchstaben geschrieben, waere hier nach "Dresden" gesucht worden, waere kein Datensatz gefunden worden.)

- Alphanumerische Ausdruecke sind in Anfuehrungszeichen (einfach oder doppelt) zu setzen, rein numerische Werte werden ohne Anfuehrungszeichen geschrieben. Ob ein Datenfeld numerisch oder alphanumerisch ist, haben Sie bereits bei der Definition der Satzstruktur festgelegt, auch hier muessen die Angaben uebereinstimmen.

- Fuer numerische und alphanumerische Daten gleichermassen gelten die aus der Mathematik bekannten Vergleichsoperatoren und deren Symbole:

=	gleich
<	kleiner
>	groesser
<>	ungleich

Ein wichtiger Unterschied besteht bei der Abfrage zwischen alphanumerischen und numerischen Zahlen. Alphanumerische Zahlen sind eine Folge einzelner Ziffern ohne Beachtung des Stellenwertes. Die Ziffernfolge 82 ist also beispielsweise groesser als die Ziffernfolge 39876, da die erste Ziffer des ersten Wertes "8" groesser ist als die erste Ziffer des zweiten Wertes "3". Das erklaert auch die Tatsache, dass es bei dem folgenden Beispiel ausreicht, fuer plz = '82' anzugeben, um alle Kunden in den Orten zu erhalten, deren Postleitzahl mit "82" beginnt.

Bild 15 demonstriert das Abfragen der Datei "kunden" mit dem Vergleichsoperator "gleich". Es soll nach Kunden in bestimmten Kreisen des Bezirkes Dresden (Postleitzahl 82xx) gesucht werden.

Bild 16 demonstriert die Anwendung des Vergleichsoperators "kleiner". Es werden alle Kunden aufgelistet, die in der Stadt Dresden wohnen. Da die Ziffernfolge "80xx" der Postleitzahl der Stadt Dresden entspricht, ist es moeglich, nach PLZ < 8100 zu fragen. Lautete die Bedingung LIST FOR plz = '80', erhielte man in diesem Beispiel dasselbe Ergebnis.

```
: LIST FOR plz = '82'

00002  Schmidt    Gisela  8223 THARANDT Bahnhofstr. 42
00003  Moser      Otto    8212 FREITAL  Neugraben 8
00004  Weber      Franz   8223 THARANDT Heideweg 66
00006  Miller     Peter   8212 FREITAL  Nelkenweg 23
00008  Neumann    Inge    8222 RABENAU  Gartenstr. 4
00009  Fleischer  Karl    8212 FREITAL  Am See 83
00010  Schmidt    Heinz   8223 THARANDT Waldweg 42
```

Bild 15 Datei "kunden", mit Vergleichsoperator "=" durchsucht

```
: LIST FOR plz < '8100'

00001  Lehmann     Kurt    8060 DRESDEN  Landstr. 15
00005  Schulze     Ernst   8060 DRESDEN  Hauptstr. 61
00012  Heinze      Max     8060 DRESDEN  Rosenstr. 78
00013  Tauber      Fred    8028 DRESDEN  Fliederstr. 42
```

Bild 16 Datei "kunden", mit Vergleichsoperator "<" durchsucht

Feldausschnitt als Suchbegriff

Mit dem sogenannten Substring-Operator kann die Datei nach beliebigen Ausschnitten innerhalb eines Datenfeldes durchsucht werden. (Substring bedeutet eigentlich Teil einer Zeichenkette.) Beispielsweise kann die Kundendatei nach allen Personen durchsucht werden, in deren Namen ein "m" vorkommt. Der LIST-Befehl wird folgendermassen formuliert, wenn man mit dem Substring-Operator arbeiten moechte:

LIST FOR <ausdruck>¤<feldname>

Handelt es sich um ein alphanumerisch definiertes Feld, ist der gesuchte Ausdruck wieder in Anfuehrungszeichen zu setzen. Danach folgt das sogenannte nationale Waehrungszeichen, entweder "¤" oder das Dollar-Zeichen, je nachdem , welches der beiden Zeichen auf Ihrer Computertastatur vorhanden ist. Zuletzt steht der Name des Feldes, in dem nach dem Ausdruck gesucht werden soll.

Bild 17 demonstriert einmal das Durchsuchen der Datei "kunden" nach Personen, in deren Name ein "m" vorkommt und zum anderen das Durchsuchen nach Namen, in denen ein grosses "M" enthalten ist, die also mit "M" beginnen.
Die beiden Befehle lauten

LIST FOR 'm'¤name und
LIST FOR 'M'¤name.

```
: LIST FOR 'm'¤name

00001  Lehmann   Kurt    8060 DRESDEN  Landstr. 15
00002  Schmidt   Gisela  8223 THARANDT Bahnhofstr. 43
00008  Neumann   Inge    8222 RABENAU  Gartenstr. 4
00010  Schmidt   Heinz   8223 THARANDT Waldweg 42

: LIST FOR 'M'¤name

00003  Moser     Otto    8212 FREITAL  Neugraben 8
00006  Miller    Peter   8212 FREITAL  Nelkenweg 23
00007  Meier     August  8122 RADEBEUL Uferstr. 12
```

Bild 17 Demonstration des "Substring"-Operators

Verknuepfen von Suchbedingungen

Will man mehrere vergleichende Bedingungen innerhalb eines Befehls formulieren, geschieht das Verknuepfen mit den drei logischen Grundfunktionen

```
.AND. (entspricht und, d.h. sowohl...als auch)
.OR.  (entspricht oder, d.h. entweder...oder)
.NOT. (entspricht nicht)
```

Diese sind jeweils in Punkte einzuschliessen. Beide im Bild 17 vorgefuehrten Befehle wuerden also zweckmaessigerweise zu dem folgenden Befehl zusammengefasst:

```
LIST FOR ('m'¤name .OR. 'M'¤name)
```

Man erhaelt eine Liste all der Personen, die entweder mit "M" beginnen oder innerhalb ihres Namens ein "m" haben (s Bild 18).

```
: LIST FOR ('m'¤name .OR. 'M'¤name)

00001  Lehmann   Kurt    8060 DRESDEN  Landstr. 15
00002  Schmidt   Gisela  8223 THARANDT Bahnhofstr. 43
00003  Moser     Otto    8212 FREITAL  Neugraben 8
00006  Miller    Peter   8212 FREITAL  Nelkenweg 23
00007  Meier     August  8122 RADEBEUL Uferstr. 12
00008  Neumann   Inge    8222 RABENAU  Gartenstr. 4
00010  Schmidt   Heinz   8223 THARANDT Waldweg 42
```

Bild 18 Zwei mit "OR" verknuepfte Bedingungen

Bild 19 zeigt ein Beispiel fuer zwei mit "AND" verknuepfte Bedingungen. Die Namen der gewuenschten Personen der Datei "kunden" sollen mit "M" beginnen und diese Kunden sollen im

Kreis Freital wohnen, d.h. die Postleitzahl muss mit den Ziffern 82 beginnen.

```
------------------------------------------------------------
|                                                          |
| : LIST FOR ('M'¤name .AND. plz = '82')                   |
|                                                          |
| 00003  Moser     Otto    8212 FREITAL  Neugraben 8       |
| 00006  Miller    Peter   8212 FREITAL  Nelkenweg 23      |
|                                                          |
------------------------------------------------------------
```

Bild 19 Zwei mit "AND" verknuepfte Bedingungen

Eindeutigkeit von Suchbedingungen

Es ist moeglich, den Substring-Operator zur Funktion zu erweitern, so dass die Position des Suchbegriffs innerhalb des Feldes genau spezifiziert werden kann:

LIST FOR <ausdruck> = ¤(<feldname>,<start>,<laenge>)

Wichtig ist das Gleichheitszeichen nach dem Suchbegriff. "start" ist diejenige Zeichenposition im Feld, von der ab verglichen werden soll, "laenge" ist ebenfalls ein Zahlenwert, der die Anzahl der zu vergleichenden Zeichen angibt.

Bild 20 zeigt, wie dieser Befehl gehandhabt wird. (Im Prinzip erhalten Sie in diesem Beispiel die gleichen Resultate, die Sie beispielsweise auch mit LIST FOR plz = '80' erhielten.) Wir moechten Sie lediglich darauf hinweisen, dass Sie bei Nichtpositionierung des Suchbegriffs "80", also mit dem Befehl

LIST FOR '80'¤plz

alle Personen faenden, die zufaellig innerhalb der vierstelligen Postleitzahl die Ziffernfolge "80" fuehren. Die Suchbedingung waere so also nicht eindeutig formuliert, denn Sie wollten nur die Kunden aufgelistet haben, die im Stadtgebiet Dresden wohnen, d.h. deren Postleitzahl mit 80 beginnt.

```
------------------------------------------------------------
|                                                          |
| : LIST FOR '80'=¤(plz,1,2)                               |
|                                                          |
| 00001  Lehmann   Kurt    8060 DRESDEN  Landstr. 15       |
| 00005  Schulze   Ernst   8060 DRESDEN  Hauptstr. 61      |
| 00012  Heinze    Max     8060 DRESDEN  Rosenstr. 78      |
| 00013  Tauber    Fred    8028 DRESDEN  Fliederstr. 42    |
|                                                          |
------------------------------------------------------------
```

Bild 20 Datei "kunden", mit positioniertem Suchbegriff durchsucht

Bedingte Suche einzelner Datensaetze

REDABAS bietet ausser dem ausfuehrlich besprochenem Befehl LIST weitere Varianten des Durchsuchens der Datei, die nachfolgend erklaert werden sollen.

LIST bringt stets alle zutreffenden Datensaetze gemeinsam zur Anzeige. Will man die Saetze jedoch einzeln aufrufen, beispielsweise um sie spaeter zu bearbeiten, geschieht das mit dem Befehl

LOCATE FOR <bedingung>

Es wird nach dem ersten Satz gesucht, der die gestellte Bedingung erfuellt. Wird ein Satz gefunden, zeigt REDABAS die Satznummer an. Der Satz selbst wird mit LOCATE nicht auf den Bildschirm gebracht. Wird die Anzeige des Satzes jedoch gewuenscht, geschieht das nach dem LOCATE-Befehl mit

DISPLAY

Der Datensatz erscheint jetzt in der gleichen Form, wie Sie es von LIST her kennen. Unabhaengig davon, ob Sie sich den Datensatz anzeigen lassen, koennen Sie mit

CONTINUE

die Suche fortsetzen und, wenn Sie es wuenschen, den naechsten gefundenen Satz wiederum mit DISPLAY anzeigen lassen.
Bild 21 demonstriert dies.

```
: LOCATE FOR 'Schmidt'¤name
SATZ:00002

: DISPLAY
00002  Schmidt   Gisela   8223 THARANDT  Bahnhofstr. 43

: CONTINUE
SATZ:00010

: DISPLAY
00010  Schmidt   Heinz    8223 THARANDT  Waldweg 42
```

Bild 21 Beispiel fuer die Anwendung der Befehle LOCATE, DISPLAY, CONTINUE

Der Befehl LOCATE FOR wird bei spezifizierten Bedingungen genau wie der Befehl LIST FOR formuliert.

3.2.3. Bedingte Anzeige von Datensaetzen

Wir moechten Ihnen den Befehl DISPLAY nachfolgend etwas ausfuehrlicher erlaeutern, da er fuer die Arbeit mit REDABAS sehr wichtig ist.

Enthaelt der Befehl DISPLAY keinerlei Zusatzangaben, bezieht er sich auf den aktuellen Satz.

Ist eine FOR-Klausel vorhanden, werden saemtliche Datensaetze, die die Bedingung erfuellen, angezeigt.

Nach jeweils 15 Datensaetzen gibt Ihnen REDABAS Gelegenheit, diese zu betrachten und erst nach Druecken der <RETURN>-Taste werden die naechsten 15 Datensaetze zur Anzeige gebracht. Hier liegt uebrigens der wesentliche Unterschied zum LIST-Befehl, der alle Informationen ohne Unterbrechung anzeigt.

DISPLAY ALL zeigt alle in der Datei vorhandenen Datensaetze an. Nach jeweils 15 Saetzen wird wieder eine Pause gemacht.

DISPLAY RECORD <zahl> zeigt den durch die Angabe der Satznummer spezifizierten Datensatz an.

DISPLAY NEXT <zahl> listet so viele der nachfolgenden Datensaetze auf, wie unter <zahl> spezifiziert wurde. Begonnen wird mit dem aufgerufenen Satz.
Zwei Formen des DISPLAY-Befehls zeigt Bild 22.

```
------------------------------------------------------------
|                                                          |
| : DISPLAY RECORD 5                                       |
| 00005  Schulze    Ernst    8060 DRESDEN   Hauptstr. 61  |
|                                                          |
| : DISPLAY NEXT 4                                         |
| 00005  Schulze    Ernst    8060 DRESDEN   Hauptstr. 61  |
| 00006  Miller     Peter    8212 FREITAL   Nelkenweg 23  |
| 00007  Meier      August   8122 RADEBEUL  Uferstr. 12   |
|                                                          |
| 00008  Neumann    Inge     8222 RABENAU   Gartenstr. 4  |
------------------------------------------------------------
```

Bild 22 Varianten des Befehls DISPLAY

3.2.4. Positionieren auf Datensaetze

Wird eine Datei mittels USE eroeffnet, wird durch REDABAS der sogenannte Datensatzzeiger stets auf die Datensatznummer 1 gestellt. Waehrend der Bearbeitung einer Datei steht dieser Datensatzzeiger dann auf dem jeweils aktivierten Datensatz. Will man von dort aus wieder an den Dateianfang zurueck, geschieht das mit dem Befehl

GO TOP

GO BOTTOM

setzt den Datensatzzeiger auf den letzten Satz. Damit kann man sehr schnell auf die zeitlich letzte Eintragung der Datei zugreifen.

GO <zahl>

positioniert den Datensatzzeiger auf jeden beliebigen Datensatz. Es genuegt auch, nur die Nummer des gewuenschten Datensatzes in der Kommandozeile anzugeben.

```
------------------------------------------------------------
|                                                          |
| : GO 6                                                   |
| : DISPLAY                                                |
| 00006  Miller    Peter    8212 FREITAL  Nelkenweg 23    |
|                                                          |
| : GO TOP                                                 |
| : DISPLAY                                                |
| 00001  Lehmann   Kurt     8060 DRESDEN  Landstr. 15      |
|                                                          |
| : GO BOTTOM                                              |
| : DISPLAY                                                |
| 00013  Tauber    Fred     8028 DRESDEN  Fliederstr. 42   |
|                                                          |
| : 8                                                      |
| : DISPLAY                                                |
| 00008  Neumann   Inge     8222 RABENAU  Gartenstr. 4     |
|                                                          |
------------------------------------------------------------
```

Bild 23 Varianten des Befehls GO

3.2.5. Abfragen einzelner Datenfelder

Sowohl mit dem Befehl LIST als auch mit dem Befehl DISPLAY ist es moeglich, einzelne Felder eines Datensatzes fuer eine Anzeige zu spezifizieren (vergl. LIST vorname, name, ort). In der Praxis wird es haeufig der Fall sein, dass nicht alle Felder eines Satzes fuer eine bestimmte Aussage gewuenscht werden bzw. wegen mangelndem Platz auf dem Bildschirm nur ein Teil der Felder angezeigt werden kann.

Eine Moeglichkeit, gezielt einzelne Datenfelder eines Datensatzes abzufragen, ist die Eingabe eines Fragezeichens mit nachfolgendem Feldnamen, wenn der Datensatz bereits im Zugriff ist oder vorher aufgerufen wurde. Bild 24 demonstriert dies.

```
------------------------------------------------
|                                              |
| : LOCATE FOR vorname = 'Otto'                |
| SATZ:00003                                   |
|                                              |
| : ? name                                     |
| Moser                                        |
| : ? plz, ort, strasse                        |
| 8212 FREITAL   Neugraben 8                   |
|                                              |
------------------------------------------------
```

Bild 24 Abfragen einzelner Felder mit dem Fragezeichen

Uebrigens koennen Sie das Fragezeichen auch nutzen, um kleine Rechnungen - aehnlich wie mit dem Taschenrechner - vorzunehmen. Sie geben nach dem Fragezeichen einfach die zu berechnenden Werte ein und erhalten in der naechsten Zeile das Ergebnis.

3.3. Aendern der Dateistruktur

In den vorangegangenen Abschnitten lernten Sie all jene Befehle kennen, mit denen Sie eine Datenbank aufbauen, pflegen und auswerten koennen.

In der praktischen Datenbankarbeit wird es aber haeufig vorkommen, dass sich aufgrund neuer Gesichtspunkte Aenderungen an vorhandenen Datenbestaenden ergeben. Das heisst, Datenfelder muessen an bestehende Dateien hinzugefuegt oder von diesen entfernt werden, konkret, die Struktur der Datei muss veraendert werden.

REDABAS bietet als relationales Datenbankbetriebssystem fuer diesen Datenbankdienst weitestgehend Datenunabhaengigkeit, d.h. die Aenderung der Dateistruktur erfolgt relativ einfach, ohne vorhandene Anwendungsprogramme zu beruehren.

Es wird folgendermassen vorgegangen:

- Die Struktur und die Daten der Datei, deren Struktur veraendert werden soll, werden in eine temporaere Datei kopiert.

- Die Dateistruktur wird veraendert.

- Anschliessend wird die neu strukturierte Datei mit den Daten aus der temporaeren Datei gefuellt.

Wir moechten Ihnen diese Vorgehensweise wiederum an unserer Datei "kunden" erlaeutert.

Fuer die Arbeit in der Praxis stellte sich heraus, dass es zweckmaessiger ist, an jeden Kunden zur eindeutigen Identifikation eine Kundennmmer zu vergeben.

Die Datei soll zunaechst eroeffnet werden:

```
USE kunden
```

Um die Daten aus der Datei "kunden" in eine temporaere Datei auszulagern, schreiben Sie den Befehl

```
COPY TO <dateiname>
```

Die gesamte Datei "kunden", d.h. ihre Struktur und die Daten werden in eine temporaere, von Ihnen zu benennende Datei kopiert.

Da es sich nur um eine zeitweilig existierende Datei handelt, nennen wir sie "temp", als Abkuerzung fuer temporaer

```
COPY TO temp
```

REDABAS meldet Ihnen nach erfolgreich durchgefuehrtem Kopieren:

```
00013 SAETZE WURDEN KOPIERT
```

Jetzt kann die Struktur der Datei "kunden" veraendert werden, ohne dass die vorhandenen Daten verloren gehen. Sie befehlen

MODIFY STRUCTURE

REDABAS warnt Sie mit der Nachricht:

MODIFY LOESCHT ALLE DATENSAETZE, TROTZDEM FORTSETZEN? (J/N)

Da Sie die Daten in die temporaere Datei kopiert haben, antworten Sie mit

J

REDABAS loescht daraufhin den Bildschirm und zeigt Ihnen die Dateistruktur in einer Editiermaske (s. Bild 25). Moechten Sie sich mehr als die auf dem Bildschirm aufgerufenen Felder anzeigen lassen, koennen Sie mittels Cursorsteuerung auf die naechste Bildschirmseite "umblaettern" und bekommt damit alle 32 moeglichen Datenfelder angezeigt.

```
-------------------------------------------------
|                                               |
|             NAME        TYP  LAE  DEZ         |
|   FELD 01 :NAME          C   010  00          |
|   FELD 02 :VORNAME       C   008  00          |
|   FELD 03 :PLZ           C   004  00          |
|   FELD 04 :ORT           C   008  00          |
|   FELD 05 :STRASSE       C   015  00          |
|   FELD 06 :                                   |
|      :                                        |
|      :                                        |
|                                               |
-------------------------------------------------
```

Bild 25 Maske fuer Aenderung der Dateistruktur

Um die von Ihnen gewuenschten Aenderungen vorzunehmen, koennen Sie den Cursor beliebig bewegen und ohne weiteres alte Eintragungen ueberschreiben. Zum Ueberschreiben uebriggebliebener Buchstaben nutzen Sie die Leertaste. Fuer die Bewegung des Cursors stehen Ihnen saemtliche Tastaturbefehle zur Verfuegung, die Sie schon kennen (vergleiche Tabelle der bildschirmorientierten Tastaturbefehle in den Abschnitten 3.1.3. und 6.2.). Erinnern Sie sich beispielsweise an <CONTROL-G> (Loeschen eines Zeichens) und <CONTROL-Y> (Loeschen der Zeile, in der der Cursor steht. Die nachfolgenden Felder ruecken nicht nach, die geleerte Zeile kann sofort beschrieben werden).

Zusaetzlich koennen Sie benutzen:

CONTROL-T Loeschen der Zeile, in der der Cursor steht, d.h. Loeschen des gesamten Feldes
Die nachfolgenden Felder ruecken nach oben auf.

CONTROL-N ermoeglicht das Einfuegen eines neuen Feldes. Alle Felder ab der Zeile, in der der Cursor steht, ruecken eine Zeile nach unten.

Wir moechten jetzt in unserem Beispiel fortfahren und in die Datei "kunden" zusaetzlich das Feld "Kundennummer" aufnehmen.

Bewegen Sie den Cursor in die erste Zeile und druecken Sie <CONTROL-N>. So koennen Sie noch vor dem Namen die Kundennummer einfuegen. Der Name des neuen Feldes soll "kunr" sein, der Typ C (alphanumerisch) und die Laenge soll drei Stellen sein.

Sie tragen also folgende Informationen in das neue Feld 01 ein:

```
         NAME       TYP  LAE  DEZ
FELD 01: KUNR        C   003
```

Mit <CONTROL-W> wird die neue Struktur gespeichert und Sie kehren zur Kommandoeingabe von REDABAS zurueck. Moechten Sie sich ueberzeugen, dass die Aenderung ordnungsgemaess erfolgt ist, geben Sie DISLAY STRUCTURE oder LIST STRUCTURE ein. Sie koennen auch mit <CONTROL-P> den Drucker zuschalten, um die neue Struktur zu protokollieren.

```
-------------------------------------------------
|                                               |
| DATEISTRUKTUR FUER:  KUNDEN  .DBD             |
| ANZAHL DER SAETZE:   00013                    |
| DATUM DER LETZTEN AENDERUNG: 04.02.85         |
| DATENBANKDATEI IM PRIMAERZUGRIFF              |
| FELD     NAME       TYP  LAENGE  DEZ          |
| 001      KUNR        C    003                 |
| 002      NAME        C    010                 |
| 003      VORNAME     C    008                 |
| 004      PLZ         C    004                 |
| 005      ORT         C    008                 |
| 006      STRASSE     C    015                 |
| ** SUMME **           00049                   |
|                                               |
-------------------------------------------------
```

Bild 26 Neue Struktur der Datei "kunden"

Nun muessen noch die Daten aus der temporaeren Datei in die neue Struktur uebertragen werden. Dies geschieht mit dem Befehl

```
APPEND FROM temp
```

Damit werden die Datenfelder zurueckgeschrieben. (Haetten Sie Felder entfernt, wuerden nur die Datenfelder zurueckgeschrieben, die in der neuen Struktur noch vorhanden sind.)

In der Datei "temp" sind alle Daten noch in der Ursprungsform vorhanden. Sie koennen sich mit

```
USE temp
LIST
```

davon ueberzeugen
Jetzt muessen Sie noch die Kundennummern in die Datei eintragen. Das koennen Sie tun, indem Sie mit dem Befehl EDIT die Datensaetze einzeln aufrufen und ergaenzen. Wir moechten Ihnen aber einen Befehl vorstellen, mit dem Sie so viele Datensaetze gleichzeitig aufrufen und bearbeiten koennen, wie auf Ihrem Bildschirm Platz haben. Dieser maechtige Befehl heisst

BROWSE

Sie koennen mit BROWSE gewissermassen "in der Datei blaettern". Die Datensaetze sind zeilenweise angeordnet und koennen wunschgemaess mit den Cursortasten und den entsprechenden Tastenkombinationen bearbeitet werden. Zusaetzlich zu den allgemeinen bildschirmorientierten Tastaturbefehlen nutzen Sie fuer den BROWSE-Befehl die folgenden Tastenkombinationen:

CONTROL-C erlaubt, den Cursor durch alle Felder mit gleichem Namen senkrecht nach unten zu bewegen.
CONTROL-R erlaubt, den Cursor durch alle Felder mit gleichem Namen senkrecht nach oben zu bewegen.

Eine wesentliche Voraussetzung fuer die Anwendung des BROWSE-Befehls ist das Einstellen des Datensatzzeigers auf den Dateianfang. Dies geschieht mit

GO TOP

Um unsere Datei "kunden" nun mit der Kundennummer zu komplettieren, geben Sie bitte ein:

```
USE kunden
GO TOP
BROWSE
```

Auf dem Bildschirm werden Ihnen die einzelnen Datensaetze zeilenweise angezeigt. Sie positionieren den Cursor auf das Feld KUNR des ersten Datensatzes und geben die Ziffer 048 ein. Bild 27 zeigt Ihnen die weiteren Kundennummern, die Sie bitte alle nacheinander eintragen. Um von dem Datenfeld eines Satzes in das gleiche Feld des naechsten zu gelangen, nutzen Sie <CONTROL-C>. Haben Sie alle Kundennummern eingetragen, schliessen Sie die Bearbeitung mit <CONTROL-W> ab und beenden damit das Kommando BROWSE.

```
------------------------------------------------------------
|                                                          |
|  SATZ- NR:00001                                          |
| KUN NAME------ VORNAME- PLZ- ORT----- STRASSE--------    |
|  48 Lehmann    Kurt     8060 DRESDEN  Landstr. 15        |
| 123 Schmidt    Gisela   8223 THARANDT Bahnhofstr. 43     |
| 645 Moser      Otto     8212 FREITAL  Neugraben 8        |
| 850 Weber      Franz    8223 THARANDT Heideweg 66        |
| 267 Schulze    Ernst    8060 DRESDEN  Hauptstr. 61       |
| 678 Miller     Peter    8212 FREITAL  Nelkenweg 23       |
| 421 Meier      August   8122 RADEBEUL Uferstr. 12        |
| 789 Neumann    Inge     8222 RABENAU  Gartenstr. 4       |
| 750 Fleischer  Karl     8212 FREITAL  Am See 83          |
| 366 Schmidt    Heinz    8223 THARANDT Waldweg 42         |
| 910 Lolle      Peter    8122 RADEBEUL Goethestr. 13      |
|  83 Heinze     Max      8060 DRESDEN  Rosenstr. 78       |
| 999 Tauber     Fred     8028 DRESDEN  Fliederstr. 42     |
|                                                          |
------------------------------------------------------------
```

Bild 27 Durch den Befehl BROWSE mit der Kundennummer komplettierte Datei "kunden"

Nachdem Sie die modifizierte Datei "kunden" ueberprueft haben, koennen Sie mit

DELETE FILE temp

die temporaere Datei jetzt loeschen.

3.4. Sortieren und Indizieren

Da Daten im allgemeinen in keiner bestimmten Reihenfolge erfasst werden, wird es fuer auszudruckende Berichte, das Verdichten oder Auswerten von Dateien oder fuer besonders schnellen Zugriff erforderlich sein, Daten durch Sortieren und Indizieren neu zu organisieren.

REDABAS bietet fuer das Sortieren einer Datei den Befehl

SORT ON <feldname> TO <dateiname> <DESCENDING>

Die aktivierte Datei wird nach dem unter "feldname" zu benennenden Datenfeld sortiert. Normalerweise erfolgt die Sortierung aufsteigend (ASCENDING), geben Sie jedoch DESCENDING an, wird absteigend sortiert. Die sortierte Datei wird in die mit "dateiname" spezifizierte Zieldatei abgespeichert, deren Name sich von der Ursprungsdatei unbedingt unterscheiden muss.

Im folgenden soll die Datei "kunden" nach aufsteigender Kundennummer sortiert werden. Nach dem Sortieren werden die Datensaetze in der temporaeren Datei "temp" zwischengespeichert, die temporaere Datei wird eroeffnet und von dieser werden die sortierten Datensaetze wieder in die Datei "kunden" uebernommen. Dafuer ist folgende Befehlsfolge erforderlich:

```
USE kunden
SORT ON kunr TO temp
USE temp
COPY TO kunden
USE kunden
LIST
```

```
-------------------------------------------------------------
|                                                           |
| 00001 048 Lehmann   Kurt    8060 DRESDEN  Landstr. 15     |
| 00002 083 Heinze    Max     8060 DRESDEN  Rosenstr. 78    |
| 00003 123 Schmidt   Gisela  8223 THARANDT Bahnhofstr. 43  |
| 00004 267 Schulze   Ernst   8060 DRESDEN  Hauptstr. 61    |
| 00005 366 Schmidt   Heinz   8223 THARANDT Waldweg 42      |
| 00006 421 Meier     August  8122 RADEBEUL Uferstr. 12     |
| 00007 645 Moser     Otto    8212 FREITAL  Neugraben 8     |
| 00008 678 Miller    Peter   8212 FREITAL  Nelkenweg 23    |
| 00009 750 Fleischer Karl    8212 FREITAL  Am See 83       |
| 00010 789 Neumann   Inge    8222 RABENAU  Gartenstr. 4    |
| 00011 850 Weber     Franz   8223 THARANDT Heideweg 66     |
| 00012 910 Lolle     Peter   8122 RADEBEUL Goethestr. 13   |
| 00013 999 Tauber    Fred    8028 DRESDEN  Fliederstr. 42  |
|                                                           |
-------------------------------------------------------------
```

Bild 28 Datei "kunden", sortiert nach der Kundennummer

Wir moechten Sie noch darauf hinweisen, dass das Zwischenspeichern in eine temporaere Datei unbedingt erforderlich ist. Eine Datei darf nie in sich selbst sortiert werden, denn eine Netzstoerung oder andere unvorhergesehene Ereignisse koennten die gesamte Datei zerstoeren. Durch das Zwischenspeichern bleibt die Ursprungsdatei so lange erhalten, bis Sie davon ueberzeugt sind, dass der Sortierablauf ordnungsgemaess ausgefuehrt wurde.

Eine neu sortierte Datei bewirkt auch eine neue Zuordnung zwischen den Datensaetzen und deren laufenden Satznummern (Im Beispiel ist der ehemalige Satz Nr.3 nach dem Sortieren z.B. Satz Nr.7 geworden). Die Satznumerierung geschieht intern durch REDABAS und ist unabhaengig vom Inhalt der Datensaetze. Dies ist besonders dann zu beachten, wenn Sie Befehle ausfuehren, die auf die Satznummern Bezug nehmen (wie z.B. EDIT).

Generell ist es moeglich, jede beliebige Sortierung einer Datei mittels temporaerer Datei vorzunehmen. Eine solche zweite, neben der "Stammdatei" laufende Datei sollte jedoch nur fuer eine einmalige Auswertung benutzt werden, um unnoetige und gefaehrliche Redundanzen in der Datenbank zu vermeiden. Man denke beispielsweise an das Problem der Aktualisierung.

Man kann jedoch mehrere Sortierordnungen gleichzeitig fuehren, indem eine Datei indiziert wird. Der Unterschied zwischen Sortieren und Indizieren besteht im folgenden:
Waehrend einer Sortierung werden die Datensaetze in der Datei tatsaechlich in ihrer Gesamtheit neu geordnet. Da die Menge der Datensaetze im allgemeinen umfangreich ist, kann es mehrere Sekunden dauern, einen bestimmten Datensatz

wiederaufzufinden.

Bei der Indizierung hingegen werden die Datensaetze in der Datei auf ihren Plaetzen belassen und es wird eine Hilfsdatei (Indexdatei) erzeugt, die pro Satz nur ein einziges Datenfeld enthaelt (den sogenannten Schluessel), nach dem die Datei sortiert ist. Ausserdem enthaelt sie zu jedem Schluesselwert einen "Zeiger" auf die Position des Datensatzes in der Datei. Dies hat den Vorteil, dass man, unabhaengig von der Groesse der Datei, sehr schnell den gesuchten Schluessel findet und man durch den Zeiger sofort den entsprechenden Datensatz in der Stammdatei wiederauffindet, ohne die gesamte Datei durchsuchen zu muessen.

Zu einer Datenbank bzw. Datei koennen auf diese Weise beliebig viele Indexdateien erstellt werden. Es ist sogar moeglich, ein bestimmtes Datenfeld auf- oder absteigend zu indexieren. Der Index-Befehl selbst lautet:

INDEX ON <ausdruck> TO <indexdatei>

Der Name der "indexdatei" ist frei waehlbar. REDABAS versieht den Namen einer Indexdatei intern mit dem Zusatz .IDX (im Unterschied zur Stammdatei, die vom Typ .DBD ist).

Um das Indexieren zu demonstrieren, erstellen wir zur Datei "kunden" eine Indexdatei "kunort", um eine Sortierung der Kunden nach Orten, d.h. eigentlich Postleitzahlen vorzunehmen. Sie geben ein:

```
USE kunden
INDEX ON plz TO kunort
```

Es erscheint die Systemnachricht:

```
00013 SAETZE INDIZIERT
```

Das Eroeffnen der Indexdatei erfolgt durch gemeinsamen Aufruf von Stammdatei und Indexdatei:

```
USE kunden INDEX kunort
```

Damit ist diese Indexdatei wie jede andere Datei verfuegbar. Bild 29 zeigt die nach den Orten indizierte Datei "kunden". Es erfolgte eine Sortierung nach Postleitzahlen und innerhalb gleicher Orte nach Kundennummern. Die Stammdatei "kunden" bleibt weiterhin allein nach Kundennummern sortiert. Man erkennt dies auch daran, dass die Satznummern noch denselben Datensaetzen wie vor der Indexierung zugeordnet sind.

Die Zuordnung der Satznummern ist wiederum unbedingt bei der Anwendung von Befehlen zu beachten, die auf diese Nummern Bezug nehmen. Das gilt beispielsweise bei Gebrauch der Befehle GO TOP oder GO BOTTOM.

GO TOP zeigt in der Datei "kunden" auf Datensatz Nr.1, in der Indexdatei auf Satz Nr.13, GO BOTTOM in der Datei "kunden" auf Satz Nr.13, in der Indexdatei auf Satz Nr.11.

```
-------------------------------------------------------------
| 00013 999 Tauber    Fred   8028 DRESDEN  Fliederstr. 42   |
| 00001 048 Lehmann   Kurt   8060 DRESDEN  Landstr. 15      |
| 00002 083 Heinze    Max    8060 DRESDEN  Rosenstr. 78     |
| 00004 267 Schulze   Ernst  8060 DRESDEN  Hauptstr. 61     |
| 00006 421 Meier     August 8122 RADEBEUL Uferstr. 12      |
| 00012 910 Lolle     Peter  8122 RADEBEUL Goethestr. 13    |
| 00007 645 Moser     Otto   8212 FREITAL  Neugraben 8      |
| 00008 678 Miller    Peter  8212 FREITAL  Nelkenweg 23     |
| 00009 750 Fleischer Karl   8212 FREITAL  Am See 83        |
| 00010 789 Neumann   Inge   8222 RABENAU  Gartenstr.4      |
| 00003 123 Schmidt   Gisela 8223 THARANDT Bahnhofstr. 43   |
| 00005 366 Schmidt   Heinz  8223 THARANDT Waldweg 42       |
| 00011 850 Weber     Franz  8223 THARANDT Heideweg 66      |
-------------------------------------------------------------
```

Bild 29 Datei "kunden", indiziert nach der Postleitzahl

Legen Sie jetzt noch eine zweite Indexdatei an, indem Sie die Stammdatei nach Kundennummern indizieren. Sie soll den Namen "ikunrkun" tragen. Geben Sie die Befehlsfolge ein:

```
USE kunden
INDEX ON kunr TO ikunrkun
USE kunden INDEX ikunrkun
```

Durch Einfuehren dieser Datei wollen wir Sie mit dem Befehl

```
FIND <zeichenreihe>
```

vertraut machen.
Mit dem FIND-Befehl laesst sich eine Indexdatei sehr schnell nach einem Suchbegriff in Form einer "zeichenreihe", die auch als Schluesselwert bezeichnet wird, abfragen. Die Suche ist deshalb so schnell, weil pro Datensatz nur ein Datenfeld (naemlich das indizierte) vorhanden ist und nicht die ganze Datei durchsucht werden muss. (Die typische Zeit fuer die Ausfuehrung eines FIND-Befehls betraegt ein bis zwei Sekunden.)

Die Suche mit dem FIND-Befehl wird beim ersten Datensatz beendet, der im indizierten Feld die "zeichenreihe" enthaelt. Der gefundene Satz kann anschliessend bearbeitet werden.

Moechten Sie beispielsweise bestimmte Aussagen ueber den Kunden mit der Kundennummer 678, so geben Sie ein:

```
FIND 678
```

Der Datensatz ist nach Erscheinen des Doppelpunktes im Zugriff und Sie koennen Anfragen stellen, wie z.B.

```
? name,ort
```

Sie erhalten als Antwort

```
Miller   FREITAL
```

Sie koennen den FIND-Befehl auch in Verbindung mit der sogenannten Makro-Funktion "&" verwenden. Auf diese Weise ist es moeglich, nach Datensaetzen zu suchen, deren Schluesselwert in einem Speicher steht.
Sie schreiben dann

FIND &<speichervariable>

FIND durchsucht die Datei stets vom Dateianfang an und endet beim ersten gefundenen Satz. Sie brauchen also nicht mit GO TOP an den Dateianfang zu gehen. Allerdings duerfen Sie die Suche auch nicht mit CONTINUE fortsetzen.

3.5. Auswerten und Verdichten

Neben den bereits bekannten Befehlen fuer den Datenbankzugriff moechten wir in diesem Abschnitt weitere Moeglichkeiten vorstellen, um Dateien auszuwerten. Dazu gehoeren Summieren, Zaehlen, das Verwenden von temporaeren Speichervariablen und das Verdichten.

Bevor wir die einzelnen Befehle beschreben, soll an dieser Stelle zusaetzlich zur bereits existierenden Datei "kunden" eine zweite Datei, genannt "auftrag" eingefuehrt werden. Diese neue Datei enthaelt Angaben zur Auftragsnummer (aufnr) und Kundennummer (kunr), ueber entstehende Betraege (betrag) das Datum der Rechnungserstellung (datum) und eine Aussage darueber, ob die Rechnung bezahlt wurde oder nicht (beza).

Da Sie mit den Befehlen fuer den Aufbau einer Datei bereits voll vertraut sind, zeigen wir Ihnen in den Bildern 30 und 31 die schon erstellte Struktur und die Datei selbst, ohne die Befehle im einzelnen zu erklaeren.

```
DATEISTRUKTUR FUER:  AUFTRAG .DBD
ANZAHL DER SAETZE:   00011
DATUM DER LETZTEN AENDERUNG: 19.02.85
DATENBANKDATEI IM PRIMAERZUGRIFF
FELD     NAME       TYP  LAENGE  DEZ
001      AUFNR       C    004
002      KUNR        C    003
003      BETRAG      N    008    002
004      DATUM       C    008
005      BEZA        L    001
** SUMME **            00025
```

Bild 30 Struktur der Datei "auftrag"

```
00001  0123 421    15.00 23.01.85 .T.
00002  0310 910    38.38 02.02.85 .F.
00003  0534 678   293.85 15.02.85 .T.
00004  1230 910    34.64 15.02.85 .T.
00005  0211 789    38.18 23.02.85 .T.
00006  2442 123   284.81 27.02.85 .F.
00007  2676 048     0.00          .F.
00008  8423 421     0.00          .F.
00009  4567 789     0.00          .F.
00010  1234 850    35.54 27.02.85 .T.
00011  6677 750  1420.00 17.02.85 .F.
```

Bild 31 Datei "auftrag"

3.5.1. Summenbildung

Wir moechten im folgenden den zu erwartenden Gesamtumsatz, den Umsatz fuer einen ausgewaehlten Monat und die Summe der Betraege der noch nicht bezahlten Auftraege ermitteln.

Fuer diese Summenbildung ueber die gesamte Datei steht der Befehl

SUM <ausdruck> FOR <bedingung>

zur Verfuegung. Es werden die Inhalte von gleichnamigen numerischen Feldern bzw. damit gebildete <ausdruecke> einer Datei summiert. Dabei ist es moeglich, bestimmte Bedingungen vorzugeben.

Zunaechst summieren wir die Betraege der Datei "auftrag", um den zu erwartenden Gesamtumsatz zu erhalten. Nachdem die Datei eroeffnet wurde, geben Sie ein

```
SUM betrag
```

und erhalten das Ergebnis:

```
2160.40
```

Es waere auch moeglich, mehrere numerische Felder zu summieren, die Feldnamen muessen hintereinander angegeben werden, untereinander durch Kommas getrennt.

Als naechstes Beispiel werden die Betraege fuer den Monat Februar errechnet.

```
SUM betrag FOR '02'=¤(datum,4,2)
2145.40
```

Nicht bezahlte Auftraege werden durch Formulieren der Bedingung "NOT" fuer das logische Feld "beza" ermittelt.

```
SUM betrag FOR .NOT. beza
1743.19
```

Es ist auch moeglich, das Ergebnis einer Summierung in einen separaten Speicher abzulegen und bei Bedarf wieder aufzurufen.

REDABAS ermoeglicht das Definieren von max. 64 sogenannten "Speichervariablen", die entweder Zahlenwerte (z.B. das Ergebnis einer Rechnung) oder alphanumerischen Text von max. 254 Zeichen Laenge beinhalten koennen. Diese Speichervariablen werden einzeln durch Vergabe eines Namens definiert und adressiert.

Unter der Ueberschrift "Verwendung temporaerer Speichervariabler") werden wir Sie noch detaillierter ueber diese Moeglichkeit von REDABAS informieren.

Um das Ergebnis einer Summierung zwischenzuspeichern formulieren wir den Befehl wie folgt

SUM <ausdruck> TO <speichervariable>

Der Name der "speichervariablen" ist frei waehlbar. Es ist jedoch sinnvoll, einerseits einen Bezug zum Feldnamen zu schaffen, andererseits sich von diesem beispielsweise durch Voranstellen eines "S" (abgeleitet von Speicher) zu unterscheiden. Die folgenden beiden Anweisungen demonstrieren das Zwischenspeichern.

```
SUM betrag TO Sbetrag
2160.40

SUM betrag FOR .NOT. beza TO Soffen
1743.19
```

Die Ergebnisse der Summierung werden sowohl auf dem Bildschirm angezeigt als auch gleichzeitig fuer die Dauer einer REDABAS-Sitzung in die entsprechenden Speicher abgelegt. Von diesen Speichern aus koennen sie ueber ihren Namen jederzeit wiederaufgerufen und weiterverarbeitet werden. Da die Speichervariablen nur im Arbeitsspeicher aufbewahrt werden (nicht auf der Diskette), muessen sie, will man sie ueber den aktuellen REDABAS-Lauf hinaus erhalten, mit SAVE fuer spaetere Programmlaeufe gesichert werden. RELEASE loescht die Speichervariablen wieder.

3.5.2. Zaehlen von Datensaetzen

Der Befehl

COUNT

zaehlt die Saetze einer Datei. Es gelten wiederum die beim Summieren beschriebenen Moeglichkeiten, Bedingungen zu formulieren und Speichervariable zu nutzen.

Formulieren Sie folgende Befehle	mit dem Ergebnis
COUNT	ZAEHLER = 00011
COUNT FOR beza	ZAEHLER = 00005
COUNT FOR betrag > 0	ZAEHLER = 00008
COUNT FOR betrag > 0 TO Sauftr	ZAEHLER = 00008

3.5.3. Verwendung temporaerer Speichervariablen

Die Speichervariablen (vgl. "Summenbildung") koennen auch direkt ueber die Tastatur Ihres Computers eingegeben werden. REDABAS stellt dafuer mehrere Befehle zur Verfuegung. Die meisten stellen wir im Abschnitt "Programmieren mit REDABAS" vor. Hier sei zunaechst auf den Befehl

STORE <ausdruck> TO <speichervariable>

verwiesen.Dieser Befehl dient dazu, erlaeuternden Text, berechnete Werte oder Konstanten, die haeufig gebraucht

werden, abzuspeichern, zu verarbeiten und wieder auszugeben. Der Wert von "ausdruck" kann numerisch oder alphanumerisch sein und wird in die zu benennende "speichervariable" eingetragen.

Es soll die Summe aller unbezahlten Auftraege ausgegeben werden und mit einer entsprechenden Ausschrift versehen werden.
Sie formulieren

```
STORE 'Summe aller unbezahlten Auftraege' TO text
? text,Soffen
```

und erhalten folgende Ausschrift auf dem Bildschirm

```
Summe aller unbezahlten Auftraege       1743.19
```

Es ist auch moeglich, Zahlenwerte als Konstanten zu speichern und wenn erforderlich, in die Rechnung einzubeziehen:

```
STORE 100 TO prozent
? Soffen/Sbetrag*prozent
```

Als Ergebnis erscheint: 80.68

Will man sich ueber die Namen und den Inhalt aller definierten Speichervariablen informieren, geschieht das mit dem Befehl

```
DISPLAY MEMORY
```

Zusaetzlich wird der Datentyp (alphanumerisch, numerisch oder logisch) angezeigt und die Summe aller definierten Speichervariablen gebildet.

Bild 32 zeigt im Ueberblick alle bis zum jetzigen Zeitpunkt erstellten Speichervariablen.

```
------------------------------------------------------------
|                                                          |
| : DISPLAY MEMORY                                         |
| SBETRAG     (N)    2160.40                               |
| SOFFEN      (N)    1743.19                               |
| SAUFTR      (N)     8                                    |
| TEXT        (C)   Summe aller unbezahlten Auftraege      |
| PROZENT     (N)     100                                  |
| ** SUMME **      05 TEMP. VARIABLE, 00062 BYTES LANG     |
|                                                          |
------------------------------------------------------------
```

Bild 32 Uebersicht der Speichervariablen

3.5.4. Verdichten von Dateien

Der Befehl

```
TOTAL ON <schluessel> TO <dateiname>
```

ermoeglicht das Verdichten von Daten. Voraussetzung fuer seine Anwendung ist, dass die aktivierte Datei nach dem "schluessel"

sortiert oder indiziert ist.

Es werden alle Saetze, die in dem spezifizierten Feld den gleichen Schluesselwert besitzen, zusammengefasst, die numerischen Felder ueber diese Saetze addiert und die zusammengefassten Saetze in eine neue Datei, die durch "dateiname" spezifiziert wird, uebernommen. Wird die Struktur der Zieldatei nicht vorher neu festgelegt, wird die Struktur der aktivierten Datei uebernommen. Diese urspruengliche Datei bleibt nach Ausfuehrung der TOTAL-Funktion voll erhalten.

Im folgenden sollen jeweils alle Auftraege eines Kunden zusammengefasst werden. Dazu wird eine nach Kundennummern indizierte Auftragsdatei eroeffnet.

```
USE auftrag
INDEX ON kunr TO ikunrauf
USE auftrag INDEX ikunrauf
TOTAL ON kunr TO gesamt
```

REDABAS erstellt die Datei "gesamt", in der alle Auftraege pro Kunde (d.h. mit gleicher Kundennummer) zusammengefasst sind. Nach Ausfuehrung des Befehls erscheint die Nachricht

```
00008 SAETZE WURDEN KOPIERT
```

Das bedeutet, dass die urspruenglich 11 vorhandenen Datensaetze auf 8 komprimiert worden sind. Pro Kunde ist nur noch ein Datensatz vorhanden, dessen numerische Feldinhalte summiert wurden.

```
-------------------------------------------------
|                                               |
| : USE gesamt                                  |
| : LIST                                        |
| 00001  2676  048        0.00           .F.   |
| 00002  2442  123      284.81  27.02.85  .F.   |
| 00003  0123  421       15.00  23.01.85  .T.   |
| 00004  0534  678      293.85  15.02.85  .T.   |
| 00005  6677  750     1420.00  17.02.85  .F.   |
| 00006  0211  789       38.18  23.02.85  .T.   |
| 00007  1234  850       35.54  27.02.85  .T.   |
| 00008  0310  910       73.02  02.02.85  .F.   |
|                                               |
-------------------------------------------------
```

Bild 33 Zusammenfassung der Auftragsdatei nach Kunden

Im Bild 33 wurde die neue Datei "gesamt" aufgelistet. Sie zeigt - geordnet nach der Kundennummer - "betrag","datum" und die Aussage ob die Betraege bezahlt oder nicht bezahlt wurden pro Kunde. Da die Struktur der Datei "gesamt" vorher nicht definiert wurde, uebernimmt REDABAS die Stuktur der Datei "auftrag" automatisch. Es zeigt sich jedoch, dass die Auftragsnummern, das logische Feld "beza" und auch das Datum keinerlei bzw. sogar verfaelschte Aussagen liefern, da diese Informationen sich ja auf mehrere Auftraege beziehen.

Es ist also sinnvoll, solche Felder zu eliminieren. Dies kann, wie bereits gesagt, von vorn herein durch Festlegen einer

neuen Struktur der Zieldatei vor der Anwendung des Befehls TOTAL geschehen und ist fuer sehr grosse Dateien auf jeden Fall zu empfehlen, um Speicherplatz und Rechenzeit einzusparen.

Bei kleineren Dateien kann man solche ueberfluessigen Felder bzw. Felder mit verfaelschtem Inhalt erst bei der Anzeige oder beim Druck unterdruecken, indem man den Befehl LIST selektiv anwendet.

Bild 34 demonstriert dies.

```
: USE gesamt
: LIST kunr,betrag
00001  048       0.00
00002  123     284.81
00003  421      15.00
00004  678     293.85
00005  750    1420.00
00006  789      38.18
00007  850      35.54
00008  910      73.02
```

Bild 34 Datei "gesamt", nachdem ueberfluessige Felder eliminiert wurden

3.6. Listengenerator

REDABAS bietet mit dem Listengenerator ein Unterprogramm, das die Aufbereitung und Ausgabe von Listen, die Nutzerwuenschen entsprechend gestaltet werden koennen, auf dem Drucker und/oder Bildschirm erlaubt, wobei das Listenformat fuer wiederholten Aufruf gespeichert wird. Von REDABAS wird eine Reihe vorprogrammierter Moeglichkeiten fuer das aeussere Schema einer Liste vorgegeben, der Nutzer gibt lediglich die entsprechenden Parameter ein. Damit wird das aufwendige Programmieren von Drucklisten eingespart. Mit dem Listengenerator haben Sie also die Moeglichkeit zu bestimmen, welche der Daten Ihrer Datei auf welche Weise aufgelistet werden sollen und ausserdem Summen und Zwischensummen zu bilden, Ueberschriften zu erstellen, die Druckseiten zu formatieren und mit Seitennummer und aktuellem Datum zu versehen. Haben Sie das Listenformular Ihren Wuenschen entsprechend aufbereitet, wird es als sogenannte Report-Datei gespeichert und ist somit jederzeit abrufbar.

Bevor wir Ihnen anhand von Beispielen die Arbeit mit dem Listengenerator demonstrieren, moechten wir zusaetzlich zu den bereits vorhandenen Dateien "kunden" und "auftrag" eine weitere Datei, genannt "bestand", einfuehren. Diese Datei soll den Lagerbestand beinhalten und enthaelt Informationen ueber die Teilnummer (teilnr) die Bezeichnung eines Teils (bezei) die Anzahl (anz) und den Preis (preis), sowie Lagerort (ortnr), Datum (datum) und den Mindestbestand (minbest).

Bild 35 zeigt die Struktur dieser Datei und Bild 36 die Datei "bestand" selbst.

```
-------------------------------------------------
|                                               |
| DATEISTRUKTUR FUER:  BESTAND .DBD             |
| ANZAHL DER SAETZE:   00020                    |
| DATUM DER LETZTEN AENDERUNG: 18.02.85         |
| DATENBANKDATEI IM PRIMAERZUGRIFF              |
| FELD     NAME       TYP  LAENGE  DEZ          |
| 001      TEILNR      C    005                 |
| 002      BEZEI       C    017                 |
| 003      ANZ         N    004                 |
| 004      PREIS       N    007    002          |
| 005      ORTNR       C    003                 |
| 006      DATUM       C    008                 |
| 007      MINBEST     N    004                 |
| ** SUMME **             00049                 |
|                                               |
-------------------------------------------------
```

Bild 35 Struktur der Datei "bestand"

```
00001 00031 Sechskantmutter   5972    0.56 110 18.01.85 2400
00002 00065 Zylinderschraube  3472    0.21 110 18.01.85 1400
00003 00106 Federring         1374    0.03 110 18.01.85 2000
00004 00108 Dichtring          766    1.90 110 18.01.85  320
00005 00327 Keilriemen         233    6.10 120 18.01.85  120
00006 00353 Scheinwerfer        30   31.60 130 11.02.85   15
00007 00425 Bremstrommel        10   14.60 140 18.01.85  120
00008 00516 Batterie             3  182.00 150 11.02.85    5
00009 00730 Lenkstock,vollst.   60   44.00 130 11.02.85   25
00010 00789 Seilzug            295    3.10 120 18.01.85  120
00011 00819 Bremsleitung        31    5.80 140 18.01.85  150
00012 00852 Bolzen             471    0.10 110 18.01.85  600
00013 14043 Ansaugkruemmer     194    2.45 140 18.01.85   80
00014 17000 Blinkleuchte, vorn 114   10.00 130 18.01.85   50
00015 30000 Bremszylinder      278    9.35 140 18.01.85  110
00016 30027 Radzylinder          6   10.50 130 18.01.85   20
00017 31100 Kolben             200   27.80 120 11.02.85   80
00018 31200 Keilriemenscheibe  253   10.90 120 18.01.85  100
00019 40018 Motor               79 1420.00 150 18.01.85   30
00020 40359 Kurbelwelle         63  280.00 140 18.01.85   30
```

Bild 36 Datei "bestand"

Der Befehl zum Aufruf des Listengenerators lautet in seiner allgemeinen Form

REPORT FORM <reportdatei>

"reportdatei" ist ein frei waehlbarer, jedoch auf den Inhalt Bezug nehmender Name, unter dem die Listenstruktur als Datei gespeichert wird und auf Wunsch wieder aufgerufen werden kann. REDABAS fuegt intern die Typenbezeichnung "DEF" an den Dateinamen an.

Es ist moeglich, den REPORT-Befehl um den Ausdruck TO PRINT zu erweitern. Dieser Zusatz bewirkt die Ausgabe der Liste auf dem Drucker, unabhaengig davon, ob der Drucker schon mit <CONTROL-P> eingeschaltet war. Da Sie Ihre Liste bestimmt immer drucken moechten, sollten Sie TO PRINT immer anfuegen.

Bevor nun der Listengenerator aufgerufen wird, muss stets die auszuwertende Datei vorher eroeffnet werden.

Es soll jetzt eine Liste des Lagerbestandes erstellt werden, wobei ausser Teilnummer, Bezeichnung und Preis der aktuelle Lagerbestand und der konstante Wert des Mindestlagerbestandes enthalten sein soll.
Mit den Befehlen

```
USE bestand
REPORT FORM best1 TO PRINT
```

wird die Datei "bestand" aufgerufen und der Name "best1" fuer eine Reportdatei vergeben.

REDABAS stellt Ihnen jetzt gezielte Fragen fuer die Gestaltung Ihrer Druckliste:

GEBEN SIE EIN, L=LINKER RAND, Z=ZEILEN/SEITE, B=SEITENBREITE

Sie brauchen die Werte aber nur eingeben, wenn Sie von der REDABAS-Standardformatierung abweichen wollen. Diese Standardwerte sind:

L	linker Rand bis zur ersten Druckposition	8 Zeichen
Z	nutzbare Druckzeilen pro Seite	57 Zeilen
B	Formularbreite bzw. Zeilenlaenge	80 Zeichen/Zeile

Die letzte Angabe (B) dient lediglich zum Zentrieren einer eventuell gewuenschten Ueberschrift und stellt keine Festlegung der maximal moeglichen Druckbreite dar.

Als naechste Frage erscheint:

WUENSCHEN SIE SEITENUEBERSCHRIFT ? (J/N):

Nach Beantwortung mit J (fuer Ja) erscheint

GEBEN SIE DIE SEITENUEBERSCHRIFT EIN:

Nachdem Sie die Ueberschrift eingetragen haben, werden Sie gefragt:

ZWEIZEILIGE AUSGABE ? (J/N):
WUENSCHEN SIE GESAMTSUMMEN ? (J/N):
WUENSCHEN SIE ZWISCHENSUMMEN IN DER LISTE ? (J/N):

Als naechstes wird die Spalteneinteilung vorgenommen. REDABAS ruft die Spalten nacheinander auf, um deren Breite und Inhalt zu definieren. Die einzelnen Spalten werden numeriert. Pro Spalte sind drei Angaben festzulegen.

Breite — Die Spaltenbreite ist hier unabhaengig von der aktuellen Laenge des Datenbankfeldes definierbar. Gibt man bei alphanumerischen Feldinhalten eine geringere Zeichenzahl ein als der eigentliche Feldinhalt betraegt, verteilt REDABAS den Inhalt auf mehrere Zeilen. Numerische Felder muessen allerdings mindestens so gross sein, dass der vorgesehene Inhalt Platz findet.

Inhalt — Der Spalteninhalt kann der Name des dafuer vorgesehenen Datenfeldes, eine temporaere Speichervariable oder ein Ausdruck, wie z.B. eine Formel sein.

Spaltenbezeichnung — Nachdem Breite und Inhalt definiert wurden, wird diese Angabe extra gefordert.
Die Spaltenbezeichnung wird in der Mitte der Spalte plaziert, wenn Sie nicht das Zeichen > fuer rechtsbuendig oder < fuer linksbuendig angeben.

Haben Sie ein numerisches Feld definiert, wird an dieser Stelle die Frage gestellt, ob fuer das Feld eine Gesamtsumme zu bilden ist:

WUENSCHEN SIE GESAMTSUMMEN ? (J/N):

Nachdem alle Angaben ordnungsgemaess eingegeben wurden, wird durch Betaetigen der <RETURN>-Taste die Definition des Formulars abgeschlossen.

Bild 37 zeigt in lueckenloser Reihenfolge alle erforderlichen Angaben fuer die Definition der Liste "Lagerbestand".

```
------------------------------------------------------------
|                                                          |
| : REPORT FORM best1 TO PRINT                             |
| GEBEN SIE EIN, L=LINKER RAND, Z=ZEILEN/SEITE, B=SEITENBR |
| EITE:L=2,Z=65,B=59                                       |
| WUENSCHEN SIE SEITENUEBERSCHRIFT? (J/N):J                |
| GEBEN SIE DIE SEITENUEBERSCHRIFT EIN:Lagerbestand        |
| ZWEIZEILIGE AUSGABE? (J/N):N                             |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):J                      |
| WUENSCHEN SIE ZWISCHENSUMMEN IN DER LISTE? (J/N):N       |
| SPALTE  BREITE, INHALT                                   |
| 001     9,teilnr                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:<Teile-Nr           |
| 002     17,bezei                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:<Bezeichnung        |
| 003     9,anz                                            |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:akt.Stand           |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):J                      |
| 004     11,minbest                                       |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:>min.Stand          |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):J                      |
| 005     10,preis                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:>Preis              |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):N                      |
| 006                                                      |
|                                                          |
------------------------------------------------------------
```

Bild 37 Dialog fuer die Erstellung einer Liste "Lagerbestand" (Reportdatei "best1")

Nachdem die Definition abgeschlossen ist, erfolgt die Verarbeitung der Datei und die Generierung der Liste. Am Anfang der Liste erscheinen Seitennummer und das von Ihnen beim Starten von REDABAS eingegebene aktuelle Datum.

Bild 38 zeigt die Liste "Lagerbestand".

SEITE 00001
11.02.85

Lagerbestand

Teile-Nr	Bezeichnung	akt.Stand	min.Stand	Preis
00031	Sechskantmutter	5972	2400	0.56
00065	Zylinderschraube	3472	1400	0.21
00106	Federring	1374	2000	0.03
00108	Dichtring	766	320	1.90
00327	Keilriemen	233	120	6.10
00353	Scheinwerfer	30	15	31.60
00425	Bremstrommel	10	120	14.60
00516	Batterie	3	5	182.00
00730	Lenkstock,vollst.	60	25	44.00
00798	Seilzug	295	120	3.10
00819	Bremsleitung	31	150	5.80
00852	Bolzen	471	600	0.10
14043	Ansaugkruemmer	194	80	2.45
17000	Blinkleuchte,vorn	114	50	10.00
30000	Bremszylinder	278	110	9.35
30027	Radzylinder	6	20	10.50
31100	Kolben	200	80	27.80
31200	Keilriemenscheibe	253	100	10.90
40018	Motor	79	30	1420.00
40359	Kurbelwelle	63	30	280.00
** SUMME **		13904	7775	

Bild 38 Mit dem Listengenerator erstellter Lagerbestand (Reportdatei "best1")

Diese obige Liste ist jetzt unter dem Namen "best1" als Reportdatei jederzeit abrufbar, beispielsweise nach dem Aktualisieren der Datei oder der Datenbank. Der Definitionsteil entfaellt dann, es sind nur die beiden Befehle zum Eroeffnen der Datei und zum Aufruf des Listengenerators erforderlich.

Allgemein ist zu sagen, dass innerhalb des Listenformulars nur der Name von Datenfeldern, aber nicht der Dateiname selbst erscheint. Das resultiert daraus, dass die Erstellung der Liste nicht an eine bestimmte Datei gebunden ist. Grundlage fuer den Listenaufbau koennen alle diejenigen Dateien sein, die die benoetigten Felder enthalten.

Auf der naechsten Liste soll der wertmaessige Lagerbestand, sortiert nach Lagerorten, ausgegeben werden. Neben den schon bekannten Spalten kommt die Spalte Gesamtwert hinzu, wobei die Feldinhalte von Preis und Anzahl (hier mit "Einzelpreis" und "Bestand" bezeichnet) miteinander multipliziert werden.

Da eine Sortierung nach Lagerorten vorgenommen werden soll, ist der Aufbau einer entsprechenden Indexdatei zweckmaessig. Auf bekannte Weise wird deshalb die Indexdatei "iortbest" erstellt und als Ausgangsdatei verwendet.

```
USE bestand
INDEX ON ortnr TO iortbest
USE bestand INDEX iortbest
```

Bild 39 zeigt die Definition der Druckliste.

```
------------------------------------------------------------
|                                                          |
| : REPORT FORM best21 TO PRINT                            |
| GEBEN SIE EIN,L=LINKER RAND, Z=ZEILEN/SEITE, B=SEITENBREI |
| TE:L=3,Z=65,B=60                                         |
| WUENSCHEN SIE SEITENUEBERSCHRIFT? (J/N):J                |
| GEBEN SIE DIE SEITENUEBERSCHRIFT EIN:  Wertmaessiger Lager|
| bestand  ;(nach Lagerorten aufgeschluesselt)             |
| ZWEIZEILIGE AUSGABE? (J/N):N                             |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):J                      |
| WUENSCHEN SIE ZWISCHENSUMMEN IN DER LISTE? (J/N):J       |
| GEBEN SIE DIE FELDNAMEN FUER ZWISCHENSUMMEN AN:ortnr     |
| LISTE NUR MIT SUMMEN? (J/N):N                            |
| SEITENVORSCHUB NACH ZWISCHENSUMMEN GEWUENSCHT? (J/N):N   |
| GEBEN SIE DIE UEBERSCHRIFTEN FUER ZWISCHENSUMMEN EIN:Lager|
| ort                                                      |
| SPALTE  BREITE, INHALT                                   |
| 001     6,teilnr                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:Teile-Nr            |
| 002     17,bezei                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:<Bezeichnung        |
| 003     7,preis                                          |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:Einzel-preis        |
| WUNSCHEN SIE GESAMTSUMMEN? (J/N):N                       |
| 004     8,anz                                            |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: Bestand            |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):N                      |
| 005     10,preis*anz                                     |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: Gesamt- wert       |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):J                      |
| 006                                                      |
|                                                          |
------------------------------------------------------------
```

Bild 39 Dialog fuer die Definition der Reportdatei "best21"

Generell gilt, dass man Textzeilen (z.B. Ueberschriften) auf mehrere Zeilen aufspalten kann. Dazu trennt man die einzelnen geplanten Zeilen durch ein Semikolon. Hier wurde die Ueberschrift "Wertmaessiger Lagerbestand" auf die erste Zeile plaziert und "nach Lagerorten aufgeschluesselt" auf die zweite. Nach dem ersten Teil der Ueberschrift muss demzufolge ein Semikolon gesetzt werden.

Im Unterschied zum vorigen Beispiel sind jetzt Zwischensummen zu bilden. Deshalb wird die entsprechende Frage jetzt mit J beantwortet, was REDABAS zu weiteren Fragen veranlasst:

GEBEN SIE DIE FELDNAMEN FUER ZWISCHENSUMMEN AN:

Da der Lagerbestand nach Lagerorten zusammengefasst werden soll, wird das Feld "ortnr" angegeben. Fuer die Zwischensummenbildung werden alle diejenigen Datensaetze erfasst, deren Feld "ortnr" den gleichen Inhalt hat.

Werden Zwischensummen gebildet, kann man entweder die Einzelposten und Zwischensummen ausgeben oder auch nur die Endsumme erscheinen lassen. Deshalb wird die Frage gestellt:

LISTE NUR MIT SUMMEN? (J/N):

Diese Frage wird fuer diese Liste mit N beantwortet, ebenso wie die naechste:

SEITENVORSCHUB NACH ZWISCHENSUMMEN GEWUENSCHT? (J/N):

Danach folgend erscheint:

GEBEN SIE DIE UEBERSCHRIFTEN FUER ZWISCHENSUMMEN EIN:

Da wir "Lagerort" angegeben haben, erscheint die Ueberschrift unmittelbar vor dem zugehoerigen Feldinhalt und kuendigt stets eine neue Gruppe an. Haette man auf diese Ueberschrift verzichtet (Druecken der <RETURN>-Taste), waere nur die Nummer des fuer die Gruppe gemeinsamen Lagerortes erschienen. Alle weiteren Ausschriften sind Ihnen bereits vertraut.

Bild 40 zeigt nun die mit der Report-Datei "best21" erzeugte Liste.

SEITE 00001
08.03.85

Wertmaessiger Lagerbestand
(nach Lagerorten aufgeschluesselt)

Teile-Nr	Bezeichnung	Einzel-preis	Bestand	Gesamt-wert
* Lagerort 110				
00031	Sechskantmutter	0.56	5972	3344.32
00065	Zylinderschraube	0.21	3472	729.12
00106	Federring	0.03	1374	41.22
00108	Dichtring	1.90	766	1455.40
00852	Bolzen	0.10	471	47.10
** ZWISCHENSUMME **				5617.16
* Lagerort 120				
00327	Keilriemen	6.10	233	1421.30
00798	Seilzug	3.10	295	914.50
31100	Kolben	27.80	200	5560.00
31200	Keilriemenscheibe	10.90	253	2757.70
** ZWISCHENSUMME **				10653.50
* Lagerort 130				
00353	Scheinwerfer	31.60	30	948.00
00730	Lenkstock,vollst.	44.00	60	2640.00
17000	Blinkleuchte,vorn	10.00	114	1140.00
30027	Radzylinder	10.50	6	63.00
** ZWISCHENSUMME **				4791.00
* Lagerort 140				
00425	Bremstrommel	14.60	10	146.00
00819	Bremsleitung	5.80	31	179.80
14043	Ansaugkruemmer	2.45	194	475.30
30000	Bremszylinder	9.35	278	2599.30
40359	Kurbelwelle	280.00	63	17640.00
** ZWISCHENSUMME **				21040.40
* Lagerort 150				
00516	Batterie	182.00	3	546.00
40018	Motor	1420.00	79	112180.00
** ZWISCHENSUMME **				112726.00
** SUMME **				154828.06

Bild 40 Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, mit Einzelpositionen (Reportdatei "best21")

Will man sehr grosse Dateien auswerten, ist es zweckmaessig, auf den Druck der einzelnen Positionen zu verzichten und nur die Zwischensummen und die Summe einer Liste auszugeben. In diesem Fall muss die Frage:

LISTE NUR MIT SUMMEN? (J/N):

mit J beantwortet werden.

Bild 41 zeigt eine solche verkuerzte Form einer Liste (Reportdatei "best22"). Gegenueber der im Bild 39 dargelegten Definition wurden folgende Veraenderungen vorgenommen: L=10, B=45; unter der Ausschrift "Spalte 001" wird 40,preis*anz eingegeben und die Spaltenueberschrift "Gesamtwert" wird rechtsbuendig gesetzt (>). Die anderen Spaltendefinitionen entfallen.

```
SEITE 00001
08.03.85

     Wertmaessiger Lagerbestand
 (nach Lagerorten aufgeschluesselt)

                                Gesamtwert

* Lagerort 110
** ZWISCHENSUMME**
                                  5617.16

* Lagerort 120
** ZWISCHENSUMME**
                                 10653.50

* Lagerort 130
** ZWISCHENSUMME**
                                  4791.00

* Lagerort 140
** ZWISCHENSUMME**
                                 21040.40
* Lagerort 150
** ZWISCHENSUMME**
                                112726.00

** SUMME **
                                154828.06
```

Bild 41 Liste "Wertmaessiger Lagerbestand", nach Lagerorten aufgeschluesselt, nur Zwischensummen und Summe (Reportdatei "best22")

Im folgenden Beispiel soll eine "Ersatzteiluebersicht" erstellt werden. Die Daten fuer diese Liste liefert die Datei "bestand", die wiederum indiziert wurde ("iortbest").

In der letzten Spalte der Liste soll der prozentuale Anteil des vorhandenen Bestandes am Mindestbestand ausgedruckt

werden. Dazu wird die Berechnungsformel

anz/minbest*100

gennommen Es sollen jedoch nur gerundete ganzzahlige Prozentwerte ausgegeben werden. Um dies zu erreichen, wird zum errechneten Wert der Wert 0.5 addiert und von dieser Summe der ganzzahlige Anteil genommen. Dafuer dient die Funktion

INT <numerischer Ausdruck>

(INT ist abgeleitet von INTEGER=ganze Zahl)
Die Spalte "in % zum Mindestbestand" wird also folgendermassen beschrieben:

006 9,INT(anz/minbest*100+0.5)

Bild 42 zeigt die Definition der Liste "Ersatzteiluebersicht" (Report-Datei "best31"), Bild 43 die Liste selbst

```
------------------------------------------------------------
|                                                          |
| : REPORT FORM best31 TO PRINT                            |
| GEBEN SIE EIN,L=LINKER RAND, Z=ZEILEN/SEITE, B=SEITENBREI |
| TE:L=0,Z=65,B=64                                         |
| WUENSCHEN SIE SEITENUEBERSCHRIFT? (J/N):J                |
| GEBEN SIE DIE SEITENUEBERSCHRIFT EIN:Ersatzteiluebersicht;|
| --------------------                                     |
| ZEIZEILIGE AUSGABE? (J/N):N                              |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):J                      |
| WUENSCHEN SIE ZWISCHENSUMMEN IN DER LISTE? (J/N):J       |
| GEBEN SIE DIE FELDNAME FUER ZWISCHENSUMMEN AN:ortnr      |
| LISTE NUR MIT SUMMEN? (J/N):N                            |
| SEITENVORSCHUB NACH ZWISCHENSUMMEN GEWUENSCHT? (J/N):N   |
| GEBEN SIE DIE UEBERSCHRIFTEN FUER ZWISCHENSUMMEN EIN:Lager|
| ort                                                      |
| SPALTE  BREITE, INHALT                                   |
| 001     6,teilnr                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:Teile-Nr            |
| 002     17,bezei                                         |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: <Bezeichnung       |
| 003     8,minbest                                        |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:Mindest-bestand     |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):N                      |
| 004     8,anz                                            |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN:>Bestand            |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):N                      |
| 005     8,anz-minbest                                    |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: Diff.-  bestand    |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):N                      |
| 006     9,INT(anz/minbest*100+0.5)                       |
| GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: in % zum  Mindest- b|
| estand                                                   |
| WUENSCHEN SIE GESAMTSUMMEN? (J/N):N                      |
| 007                                                      |
|                                                          |
------------------------------------------------------------
```

Bild 42 Dialog zur Definition der Liste "Ersatzteil-uebersicht" (Reportdatei "best31")

SEITE 00001
13.03.85

Ersatzteiluebersicht

Teile-Nr	Bezeichnung	Mindest-bestand	Bestand	Diff.-bestand	in % zum Mindest-bestand
* Lagerort 110					
00031	Sechskantmutter	2400	5972	3572	249
00065	Zylinderschraube	1400	3472	2072	248
00106	Federring	2000	1374	-626	69
00108	Dichtring	320	766	446	239
00852	Bolzen	600	471	-129	79
* Lagerort 120					
00327	Keilriemen	120	233	113	194
00798	Seilzug	120	295	175	246
31100	Kolben	80	200	120	250
31200	Keilriemenscheibe	100	253	153	253
* Lagerort 130					
00353	Scheinwerfer	15	30	15	200
00730	Lenkstock,vollst.	25	60	35	240
17000	Blinkleuchte,vorn	50	114	64	228
30027	Radzylinder	20	6	-14	30
* Lagerort 140					
00425	Bremstrommel	120	10	-110	8
00819	Bremsleitung	150	31	-119	21
14043	Ansaugkruemmer	80	194	114	243
30000	Bremszylinder	110	278	168	253
40359	Kurbelwelle	30	63	33	210
* Lagerort 150					
00516	Batterie	5	3	-2	60
40018	Motor	30	79	49	263

Bild 43 Liste "Ersatzteiluebersicht" (Reportdatei "best31")

Man kann den Befehl REPORT FORM auch in Abhaengigkeit von Bedingungen formulieren, die wieder mit dem Zusatz FOR eingeleitet werden. Damit ist es moeglich, auch fuer die Formulierung einer Liste alle bekannten Selektionsmoeglichkeiten anzuwenden. Die FOR-Klausel ist nicht Bestandteil der Listenstruktur und kann erst beim Abruf der Liste an den REPORT-Befehl angehangen werden. Damit ist es moeglich, mit einer einzigen Listenstruktur mehrere verschiedene Listen zu erstellen.

Es soll im folgenden - als letztes Beispiel der Anweisung des Listengenerators - eine Bedarfsliste erstellt werden. Bedarf an Teilen besteht dann, wenn der tatsaechliche Bestand kleiner als der geforderte Mindestbestand ist.

Bevor der REPORT-Befehl mit der entsprechenden Bedingung formuliert wird, wird ein neuer Befehl eingefuegt, um eine zweite erlaeuternde Ueberschrift in die erste Zeile der Liste zu plazieren. Dieser lautet:

```
SET HEADING TO ...        (Setze die Ueberschrift)
```

Diese zweite Ueberschrift soll "Bedarfsliste" lauten.
Fuer die Liste ist folgende Befehlsfolge ausreichend:

```
USE bestand INDEX iortbest
SET HEADING TO Bedarfsliste
REPORT FORM best31 FOR anz<minbest
```

Bild 44 zeigt die "Bedarfsliste".

SEITE 00001 Bedarfsliste
14.03.85

Ersatzteiluebersicht

Teile-Nr	Bezeichnung	Mindest-bestand	Bestand	Diff.-bestand	in % zum Mindest-bestand
* Lagerort 110					
00106	Federring	2000	1374	-626	69
00852	Bolzen	600	471	-129	79
* Lagerort 130					
30027	Radzylinder	20	6	-14	30
* Lagerort 140					
00425	Bremstrommel	120	10	-110	8
00819	Bremsleitung	150	31	-119	21
* Lagerort 150					
00516	Batterie	5	3	-2	60

Bild 44 Bedarfsliste

3.7. Gleichzeitiges Arbeiten mit mehreren Dateien

Redundanzarmes Speichern, das zu den Schwerpunkten der Datenbanktechnologie gehoert, erfordert einen flexiblen Datenbankzugriff. So kommt es darauf an, Daten, die nur einmal in der Datenbank gespeichert sind, von unterschiedlichen Programmen aus anzusprechen und zu bearbeiten und sie gegebenenfalls aus verschiedenen Dateien zusammenzufuehren.

Fuer das gleichzeitige Bearbeiten von zwei Dateien und das Erzeugen einer neuen Datei aus zwei bestehenden bietet REDABAS als relationales Datenbankbetriebssystem sehr leistungsfaehige Befehle, die nachfolgend vorgestellt werden.

3.7.1. Gleichzeitiges Bearbeiten von zwei Dateien mit unterschiedlicher Dateistruktur

Bevor diese Moeglichkeit demonstriert wird, soll eine neue Datei mit dem Namen "aufpos" erstellt werden. Diese Datei soll saemtliche Positionen pro Auftrag erfassen und soll die Felder "aufnr" (Auftragsnummer), "teilnr" (Teilnummer), "anz" (Anzahl) und "betrag" enthalten. Die Stuktur der Datei ist in Bild 45 dargestellt, die Datei selbst in Bild 46.

```
DATEISTRUKTUR FUER:  AUFPOS  .DBD
ANZAHL DER SAETZE:   00005
DATUM DER LETZTEN AENDERUNG: 25.03.85
DATENBANKDATEI IM PRIMAERZUGRIFF
FELD     NAME       TYP  LAENGE  DEZ
001      AUFNR       C    004
002      TEILNR      C    005
003      ANZ         N    004
004      BETRAG      N    008    002
** SUMME **            00022
```

Bild 45 Struktur der Datei "aufpos"

```
: USE aufpos
: LIST
00001  4567 40359        1            0.00
00002  4567 14043        1            0.00
00003  4567 00106        4            0.00
00004  4567 00031        4            0.00
00005  8423 40018        1            0.00
```

Bild 46 Datei "aufpos"

Sie erinnern sich sicher, dass in der Datei "bestand" u.a. die Information ueber den Preis eines Teiles enthalten ist. Um die Kosten der Teile eines Auftrages und dessen Gesamtkosten zu ermitteln, ist es also erforderlich, aus dieser Bestandsdatei den Einzelpreis pro Teil zu entnehmen, aus der Datei "aufpos" die Anzahl der Teile pro Auftrag und beide Werte miteinander zu multiplizieren. Es ist zweckmaessig, die Datei "bestand" nach der Teilnummer zu indizieren, um mit dem Befehl FIND schnell auf bestimmte Datensaetze dieser Datei zugreifen zu koennen. Die Datei "bestand" und die Datei "aufpos" muessen gleichzeitig im Zugriff sein.

Fuer die gleichzeitige Bearbeitung von zwei Dateien bietet REDABAS folgende Moeglichkeiten. Die Dateien werden als "primaer" und "sekundaer" klassifiziert. Ist bereits eine Datei eroeffnet, wird diese intern stets als primaere Datei betrachtet.

Im Gegensatz zum bisher bekannten "einfachen Eroeffnen" mit USE, bei dem die jeweils eroeffnete Datei nach Eroeffnen einer zweiten Datei wieder geschlossen wird und damit nicht mehr ansprechbar ist, bleibt nach dem Befehl

SELECT SECONDARY

und folgendem

USE

die vorher mit USE eroeffnete erste Datei ebenfalls verfuegbar. Beide Dateien sind fuer eine Bearbeitung bereit. Der Datensatzzeiger fuer die erste Datei verbleibt auf dem zuletzt bearbeiteten Datensatz. Die zweite Datei erhaelt einen eigenen Datensatzzeiger, der auf dem aktuellen Datensatz steht.

Die beiden Datein koennen abwechselnd bearbeitet werden. Verwendet man

SELECT PRIMARY

kann die erste Datei angesprochen werden, verwendet man

SELECT SECONDARY

wird die zweite Datei bearbeitet.
Allerdings ist zu beachten, dass man sich stets am zuletzt bearbeiteten Datensatz befindet (Im Gegensatz zu USE, nach dessen Verwendung der Datensatzzeiger immer auf den ersten Satz der Datei gerichtet ist.)

Das eben Gesagte soll nachfolgend am Beispiel demonstriert werden.

Um die Kosten der Teile pro Auftrag sowie die Gesamtkosten pro Auftrag zu ermitteln und diese dann in der Datei "aufpos" abzuspeichern, sind folgende Schritte erforderlich:
Die Datei "aufpos" wird als Primaerdatei eroeffnet. Es wird eine Indexdatei der Datei "bestand" namens "iteilbes" (nach "teilnr" indiziert) erstellt, die zusammen mit der

Bestandsdatei als Sekundaerdatei wirken soll.

```
USE bestand
INDEX ON teilnr TO iteilbes
USE aufpos
SELECT SECONDARY
USE bestand INDEX iteilbes
```

Damit sind sowohl Primaerdatei ("aufpos") und Sekundaerdatei ("bestand" und "iteilbes") eroeffnet. Aus beiden Dateien werden nun die Teile mit gleicher Teilnummer aufgesucht. Die Teilnummer aus der Primaerdatei wird in eine Speichervariable, genannt "Steil" gespeichert und mit dem sehr schnellen FIND-Befehl (vgl. Abschnitt 3.4.) wird die Sekundaerdatei auf uebereinstimmende Teilnummer durchsucht:

```
SELECT PRIMARY
STORE teilnr TO Steil
SELECT SECONDARY
FIND &Steil
```

Beide Teilnummern stimmen ueberein. Nachdem der Preis (aus der Datei "bestand") mit der Anzahl der Teile (aus der Datei "aufpos") multipliziert wird, soll er in das Feld "betrag" der Datei "aufpos" gebracht werden. Dazu dient der Befehl REPLACE (vgl. Abschnitt 3.1.3.). Das vorher leere Feld "betrag" wird jetzt mit dem Produkt der Multiplikation gefuellt.

Fuer die Arbeit mit mehreren Dateien muss natuerlich definiert werden, aus welchen Dateien die angesprochenen Felder zu entnehmen sind. Feldern aus der Primaerdatei wird ein "P." vorangestellt, Feldern aus der Sekundaerdatei ein "S.". Das ist nicht unbedingt erforderlich fuer eine Datei, die gerade mit SELECT spezifiziert wurde bzw. dann, wenn die Feldnamen nur in einer der beiden Dateien vorkommen. Wir setzen die Befehlsfolge fort:

```
SELECT PRIMARY
REPLACE P.betrag WITH S.preis * P.anz
```

Als Systemnachricht erhalten Sie von REDABAS:

```
00001 ERSETZUNG(EN)
```

Nachdem nun die Kosten der ersten Position eines Auftrages berechnet wurden, gehen wir zur naechsten Auftragsposition ueber. Das heisst, wir muessen den naechsten Satz der Primaerdatei auf die gleiche Weise bearbeiten.
Mit dem Befehl

```
SKIP
```

wird zum naechsten Datensatz uebergegangen und die Befehlsfolge fortgesetzt mit

```
STORE teilnr TO Steil
SELECT SECONDARY
FIND &Steil
SELECT PRIMARY
```

```
REPLACE P.betrag WITH S.preis * P.anz
SKIP
  .
  .
  .
```

Die Befehlsfolge muss so lange fortgesetzt werden, bis der letzte Datensatz der Primaerdatei bearbeitet wurde.

Sie werden sicherlich bemerkt haben, dass diese Art der Bearbeitung, d.h. fuer jeden Datensatz die gleiche Befehlsfolge ablaufen zu lassen, keinesfalls effektiv ist. Wir wollten Ihnen anhand des vorangehenden Beispiels lediglich die Wirkungsweise der gleichzeitigen Bearbeitung von zwei Dateien demonstrieren, im naechsten Abschnitt "Programmieren" werden Sie kennenlernen, wie derartig gleichartigen Befehlsfolgen als Programmschleifen ausgebildet werden koennen.
Im Bild 47 ist die eben beschriebene Befehlsfolge - ausser der Indizierung der Bestandsdatei - noch einmal dargestellt.

```
: USE aufpos
: SELECT SECONDARY
: USE bestand INDEX iteilbes
: SELECT PRIMARY
: STORE teilnr TO Steil
: SELECT SECONDARY
: FIND &Steil
: SELECT PRIMARY
: REPLACE P.betrag WITH S.preis * P.anz
: SKIP
: STORE teilnr TO Steil
: SELECT SECONDARY
: FIND &Steil
: SELECT PRIMARY
: REPLACE P.betrag WITH S.preis * P.anz
: SKIP
:
```

Bild 47 Befehlsfolge fuer das gleichzeitige Bearbeiten von zwei Dateien mit unterschiedlicher Struktur

Bild 46 zeigte die Datei "aufpos" vor der Bearbeitung, d.h. das Feld "betrag" war bei allen Datensaetzen ohne Eintragung. Bild 48 zeigt die nunmehr bearbeitete Datei "aufpos". Das Feld "betrag" enthaelt die Kosten gleicher Teile pro Auftrag.

```
: USE aufpos
: LIST
00001  4567 40359        1        280.00
00002  4567 14043        1          2.45
00003  4567 00106        4          0.12
00004  4567 00031        4          2.24
00005  8423 40018        1       1420.00
```

Bild 48 Datei "aufpos" nach der Bearbeitung

Fuer den Nutzer wird es natuerlich wichtig sein, nicht nur die Kosten der Teile pro Auftrag zu erfahren, sondern den Gesamtpreis (d.h. die Kosten aller Teile) pro Auftrag.

Unter der Ueberschrift "Verdichten von Dateien" lernten Sie bereits den Befehl TOTAL kennen, mit dem alle Saetze, die in einem spezifizierten Feld den gleichen Schluesselwert besitzen, zusammengefasst, addiert und in eine neue Datei uebernommen werden koennen.

Wir verdichten also die Datei "aufpos" indem wir die Teile, die zur gleichen Auftragsnummer gehoeren, zusammenfassen. Die Datei, in die die so behandelten Datensaetze uebernommen werden, soll "rech" heissen und lediglich die Felder "aufnr" (Auftragsnummer), "betrag" und zusaetzlich ein Feld "datum" enthalten. Die Dateistruktur wird vor der Anwendung des Befehls TOTAL definiert und ist in Bild 49 zu sehen.

```
------------------------------------------------
|                                              |
| DATEISTRUKTUR FUER:  RECH    .DBD            |
| ANZAHL DER SAETZE:   00002                   |
| DATUM DER LETZTEN AENDERUNG: 21.03.85        |
| DATENBANKDATEI IM PRIMAERZUGRIFF             |
| FELD     NAME       TYP  LAENGE  DEZ         |
| 001      AUFNR       C    004               |
| 002      BETRAG      N    008    002        |
| 003      DATUM       C    008               |
| ** SUMME **             00021                |
|                                              |
------------------------------------------------
```

Bild 49 Struktur der Datei "rech"

Bevor die Datei "aufpos" verdichtet wird, muss sie nach der Auftragsnummer indiziert werden. Auf bekannte Weise wird eine Indexdatei namens "inaufpos" erstellt, und anschliessend werden die folgenden Befehle eingegeben:

```
USE aufpos
INDEX ON aufnr TO inaufpos
USE aufpos INDEX inaufpos
TOTAL ON aufnr TO rech
```

Sie erhalten von REDABAS die Nachricht:

```
00002 SAETZE WURDEN KOPIERT
```

Wir listen die Datei "rech" auf und erhalten folgendes Bild:

```
---------------------------------------
|                                     |
| 00001  4567    284.81               |
| 00002  8423   1420.00               |
|                                     |
---------------------------------------
```

Bild 50 Datei "rech"

Die so entstandene Datei "rech" ist an sich nur von temporaerer Bedeutung, denn die Informationen ueber einen Auftrag bezueglich Auftragsnummer, Kundennummer, Betrag, Datum und der Aussage , ob die Rechnung bezahlt wurde oder nicht, sind in der Datei "auftrag" gespeichert (vgl. Abschnitt 3.5.) und werden vom Nutzer letztendlich von dort wiederaufgerufen.

Es kommt darauf an, die Daten aus der Datei "rech" in die Datei "auftrag" zu uebertragen. Dabei ist zu beachten, dass es sich nicht um neue, zusaetzliche Datensaetze handelt, sondern dass Datensaetze mit gleicher Auftragsnummer in der Datei "auftrag" bereits existieren. Diese Uebertragung bedeutet also eine Aktualisierung vorhandener Informationen.

Fuer das Aktualisieren bietet REDABAS den Befehl

UPDATE FROM <dateiname> ON <schluessel> ADD/REPLACE <feldnamen>

Dieser Befehl ermoeglicht das Ersetzen und/oder Ergaenzen einer Datei durch eine andere Datei Satz fuer Satz. Unter "feldnamen" ist anzugeben, welche Felder durch den Inhalt der zweiten Datei ersetzt werden sollen (REPLACE) bzw. zu welchen Feldinhalten der Inhalt entsprechender Felder der zweiten Datei hinzugefuegt werden soll (ADD). Dies wird dann realisiert, wenn die Inhalte des als "schluessel" definierten Feldes in beiden Dateien uebereinstimmen. Dabei ist es erforderlich, dass beide Dateien nach dem angegebenen "schluessel" aufsteigend sortiert oder indiziert sind, da nach der Positionierung auf den Anfang abwechselnd beide Dateien satzweise verglichen werden. So ist gewaehrleistet, dass saemtliche Datensaetze der zu aktualisierenden Datei ueberprueft und gegebenenfalls aktualisiert werden.

Anmerkung: Der Befehl UPDATE bietet ueber das hier Gesagte (und fuer dieses Beispiel Wesentliche) weitere Moeglichkeiten. Deshalb sei an dieser Stelle auf die ausfuehrliche Beschreibung von UPDATE im Abschnitt 8.3. verwiesen.

Verfolgen wir wieder unser Beispiel. Die Datei "auftrag" wird nach Auftragsnummern indiziert und zusammen mit der neugebildeten Indexdatei "iaufnrau" eroeffnet.

```
USE auftrag
INDEX ON aufnr TO iaufnrau
USE auftrag INDEX iaufnrau
UPDATE FROM rech ON aufnr REPLACE betrag
```

Beide Dateien sind jetzt nach der Auftragsnummer sortiert (durch Einfuehren der Indexdatei "iaufnrau"), der Datensatzzeiger weist jeweils auf den ersten Datensatz. Die Datei "rech" wird nach der Auftragsnummer durchsucht, die im ersten Satz der Datei "auftrag" steht (im Beisp. Auftragsnummer 0123). Da diese dort nicht vorhanden ist, wird zum naechsten Satz weitergegangen usw. Ist Auftragsnummer 4567 erreicht, wird diese in beiden Dateien gefunden, und die Daten werden aktualisiert. Danach wird weiter durchsucht, bis die Auftragsnummer 8423 erreicht wird und wiederrum uebereinstimmende Auftragsnummern erkannt werden. Entsprechend erfolgt die Aktualisierung. Es sind keine weiteren

uebereinstimmenden Auftragsnummern vorhanden, damit ist die Aktualisierung abgeschlossen. Bild 51 zeigt die aktualisierte Datei "auftrag".

```
-------------------------------------------------
|                                               |
| 00001  0123 421        15.00 23.01.85 .T.     |
| 00002  0310 910        38.38 02.02.85 .F.     |
| 00003  0534 678       293.85 15.02.85 .T.     |
| 00004  1230 910        34.64 15.02.85 .T.     |
| 00005  0211 789        38.18 23.02.85 .T.     |
| 00006  2442 123       284.81 27.02.85 .F.     |
| 00007  2676 048         0.00          .F.     |
| 00008  8423 421      1420.00          .F.     |
| 00009  4567 789       284.81          .F.     |
| 00010  1234 850        35.54 27.02.85 .T.     |
| 00011  6677 750      1420.00 17.02.85 .F.     |
-------------------------------------------------
```

Bild 51 Datei "auftrag" nach der Aktualisierung

3.7.2. Verknuepfen von zwei Dateien

Fuer das Verknuepfen von zwei Dateien zu einer dritten bietet REDABAS einen sehr leistungsfaehigen Befehl, der in seiner allgemeinen Form lautet:

JOIN TO <dateiname> FOR <bedingung> FIELD <feldnamen>

Die beiden Ursprungsdateien werden mit SELECT PRIMARY als Primaerdatei und SELECT SECONDARY als Sekundaerdatei eroeffnet. Der Datensatzzeiger weist in beiden Dateien zunaechst auf den ersten Satz. Der erste Satz der Primaerdatei wird gemaess der FOR-Bedingung mit jedem Satz der im Sekundaerbereich aktivierten Datei ausgewertet. Jeder Satz, der dieser Bedingung entspricht, wird in die mit "dateiname" spezifizierte Datei geschrieben. Es muss unbedingt darauf hingewiesen werden, dass bei dem Befehl JOIN die FOR-Klausel nicht wahlfrei, sondern zwingend ist. Sie entscheidet darueber, welche Saetze der ersten Datei mit welchen Saetzen der zweiten Datei zu neuen Datensaetzen der dritten Datei kombiniert werden.

Anders sieht es bei der FIELD-Klausel aus. Diese ist wahlfrei. Wird sie nicht benutzt, erhalten die Saetze der neuen, dritten Datei die Eintragungen der Saetze der Primaerdatei und gegebenenfalls bis zur maximal moeglichen Anzahl von 32 Feldern aus den Saetzen der Sekundaerdatei (entsprechend der dort vorhandenen Reihenfolge).

Wird die FIELD-Klausel verwendet, werden nur die unter "feldnamen" spezifizierten Eintragungen in die dritte Datei uebernommen.

Sind die anzusprechenden Feldnamen nur in einer der beiden Ursprungsdateien vorhanden, spricht REDABAS von selbst die richtige Datei an. Kommen jedoch gleiche Feldnamen in beiden Dateien vor, muss spezifiziert werden, aus welcher Datei der

Feldinhalt zu nehmen ist. Felder der Primaerdatei erhalten vor dem Feldnamen den Vorsatz "P.", Felder der Sekundaerdatei den Vorsatz "S.".

Es ist zu beachten, dass der Befehl JOIN bei grossen Dateien viel Zeit zur Ausfuehrung benoetigt. Ausserdem besteht die Gefahr, dass bei einer ungeeigneten bzw. zu grosszuegig gehandhabten Auswahl der FOR-Klausel die Grenzwerte der Dateigroesse ueberschritten werden. Wuerden z.B. mit JOIN zwei Dateien mit je 1000 Saetzen bei stets erfuellter FOR-Bedingung vereinigt, muesste die dritte Datei 1 000 000 Datensaetze aufnehmen. Eine Datei kann aber maximal nur 65 535 Saetze aufnehmen. Daher muss bei der Vereinigung von Dateien groesste Sorgfalt auf die Auswahl einer geeigneten Bedingung gelegt werden.

Der Befehl JOIN muss im folgenden Beispiel wegen seiner Laenge ueber zwei Zeilen geschrieben werden. Im Abschnitt 2.3. erfuhren Sie bereits, dass dies moeglich ist, wenn man die erste Zeile mit einem Semikolon abschliesst, anschliessend die <RETURN>-Taste drueckt und in der naechsten Zeile weiterschreibt. Die Wirkungsweise des Befehls JOIN soll nachfolgend an einem Beispiel demonstriert werden.

In eine Datei namens "kundrech" sollen die Kunden namentlich mit ihrer vollstaendigen Adresse, dem Betrag der noch zu bezahlenden Rechnung und deren Erstellungsdatum aufgenommen werden. Die Informationen sind aus der Datei "auftrag" (Primaer) und der Datei "kunden" (Sekundaer) zu entnehmen. Es werden aus beiden Dateien jeweils die Saetze vereinigt, die eine gemeinsame Kundennummer aufweisen.
Die entsprechende Befehlsfolge lautet:

```
USE auftrag
SELECT SECONDARY
USE kunden
JOIN TO kundrech FOR (P.kunr = S.kunr .AND. .NOT. beza .AND.;
betrag>0) FIELD vorname,name,plz,ort,strasse,betrag,datum
```

Bedingungen fuer die Schaffung neuer Datensaetze sind neben gleichen Kundennummern, dass schon Rechnungsbetraege vorliegen, diese aber noch nicht bezahlt sind. In diesem Beispiel wurde also die Bildung neuer Datensaetze von mehreren Bedingungen gleichzeitig abhaengig gemacht. Damit wurde die Moeglichkeit demonstriert, den Befehl JOIN selektiv anzuwenden. Im Beispiel ergeben sich fuenf Datensaetze, die den genannten Bedingungen entsprechen, sie sind in Bild 52 aufgelistet.

```
Peter  Lolle     8122 RADEBEUL Goethestr.13    38.38 02.02.85
Gisela Schmidt   8223 THARANDT Bahnhofstr.43  284.81 27.02.85
August Meier     8122 RADEBEUL Uferstr.12    1420.00
Inge   Neumann   8222 RABENAU  Gartenstr. 4   284.81
Karl   Fleischer 8212 FREITAL  Am See 83     1420.00 17.02.85
```

Bild 52 Mittels JOIN gebildete Datei "kundrech"

3.8. Zusammenfassung

Im Verlauf der vorangegangenen Abschnitte haben Sie ´die wichtigsten Methoden fuer den Aufbau und die Bearbeitung einer Datenbank mit REDABAS kennengelernt. Sie kennen damit alle wesentlichen Befehle, um Dateien zu erstellen, Daten zu laden, zu aendern und zu loeschen sowie Dateien auszuwerten. Sie wissen auch, wie bestehende Dateien bezueglich ihrer Struktur veraendert, sortiert oder indiziert und untereinander verknuepft werden. Sie haben erfahren, wie Sie Ihre Auswertungen, Berichte und Listen mit dem REDABAS-Listengenerator problemlos gestalten und ausgeben koennen. Die einzelnen Befehle wurden erklaert und anhand von Beispielen vorgefuehrt. (Eine detaillierte Beschreibung aller REDABAS-Befehle und -Funktionen finden sie in den Abschnitten 8.3. und 7.4.)

Mit diesen Informationen ausgeruestet, sind Sie in der Lage, Ihr Datenbankproblem auf einem Personalcomputer zu loesen. Nochmals sei auf die Uebersichtlichkeit und Leistungsfaehigkeit der Loesungswege mit REDABAS als relationales Datenbankbetriebssystem verwiesen.

REDABAS verfuegt aber ueber diese Moeglichkeiten hinaus noch ueber weitere Befehle, die es Ihnen erlauben, Anwendungsprogramme zu schreiben, ohne das Sie herkoemmliche Programmiersprachen erlernen oder anwenden brauchen. So koennen Sie die Befehle, mit denen Sie jetzt vertraut sind, zu Programmen zusammenstellen, diese in Dateien abspeichern und bei Bedarf wieder aubrufen Diese und weitere Moeglichkeiten der Programmierung mit REDABAS lernen Sie im Abschnitt 4. dieser Programmtechnischen Beschreibung kennen.

4. Programmieren

Wir moechten Sie in diesem Abschnitt mit den Moeglichkeiten von REDABAS bekanntmachen, Befehlsfolgen in Dateien abzuspeichern, wo sie jederzeit abrufbar sind. Sie lernen die entsprechenden Befehle kennen und darueber hinaus Mittel fuer die Bildung von Programmschleifen und Programmverzweigungen.

Damit steht Ihnen im Rahmen von REDABAS eine sehr leistungsfaehige Programmiersprache zur Verfuegung.

Die Wirkungsweise der einzelnen Befehle wird wieder an Beispielen demonstriert, die auf den aus dem Abschnitt 3. bekannten Dateien "kunden", "bestand", "auftrag" und "aufpos" basieren.

Fuer Ihre jetzige Arbeit mit diesen Dateien wird es notwendig werden, zusaetzliche Daten aufzunehmen und zu bearbeiten. Um aber ebenso jederzeit auf den urspruenglichen Zustand zurueckgreifen zu koennen, empfiehlt es sich, von den Dateien Duplikate anzufertigen. Diese sogenannten "Backup"-Kopien werden unkompliziert auf folgende Weise erstellt:

```
USE kunden
COPY TO kundbak
```

Genauso verfahren Sie mit den anderen Dateien. Der Zuzatz "bak" soll auf "Backup"-Kopie verweisen. Indexdateien brauchen nicht kopiert werden, sie werden durch Indizieren der Stammdatei je nach Bedarf erzeugt.

4.1. Befehlsfolgen

Die abzuspeichernde Befehlsfolge soll in eine Befehlsdatei (engl. COMMAND FILE) geschrieben werden, damit sie - einmal dort abgespeichert - jederzeit unter dem vergebenen Programmnamen wieder abgearbeitet werden kann.

Die Eroeffnung der Befehldatei erfolgt mit

```
MODIFY COMMAND <dateiname>
```

Der Dateiname (wiederum aus max. 8 Zeichen bestehend und mit einem Buchstaben beginnend) wird intern mit dem Zusatz .PRG durch REDABAS versehen.

Wenn Sie nachfolgend Ihre Programmdatei namens "prog1" mit

```
MODIFY COMMAND prog1
```

eroeffnet, wird der gesamte Bildschirm geloescht, der Cursor befindet sich auf dem leeren Bildschirm links oben. Sie koennen jetzt Ihre Befehle nacheinander eingeben.

Sie befinden sich im sogenannten Programmeditor, einem Hilfsprogramm zum Erstellen und Aendern von Anwendungsprogrammen. Sie koennen aber Ihr Programm auch mit einem verfuegbaren Texteditor erstellen. Arbeiten Sie beispielsweise mit dem Textverarbeitungssystem TEXT 30, ist das Schreiben von Programmen auch mit diesem System moeglich. Sie geben in diesem Fall der Textdatei selbst den Programmnamen und muessen auch den Dateityp vorschreiben. In diesem Falle: prog1.PRG

Bevor Sie nun eine Befehlsfolge abspeichern, noch einige allgemeine Hinweise fuer das Schreiben des Programms .Beginnen Sie das Programm mit dem Befehl

ERASE

Damit wird grundsaetzlich bei Aufruf des Programms als erstes der Bildschirm geloescht und der Cursor in die linke obere Ecke plaziert. ERASE kann, wie die anderen Befehle auch, einzeln benutzt werden.

Wenn Sie die einzelnen Befehle des Programms schreiben, druecken Sie, wie gewohnt, die <RETURN>-Taste nach jeder Programmzeile.

Beenden des Programms und Rueckkehr in die vorherige Kommandoebene von REDABAS realisiert der Befehl

RETURN

Mit dem Befehl RETURN muessen Sie Ihre abzuspeichernde Befehlsfolge also stets abschliessen.

Fuer das Eingeben der Befehle nutzen Sie wieder die schon bekannten bildschirmorientierten Tastatur-Befehle. Besonders wichtig sind jetzt neben den Tasten zur Cursorsteuerung die Tastenkombinationen

CONTROL-G	fuer das Loeschen einzelner Zeichen
CONTROL-T	fuer das Loeschen von Zeilen
CONTROL-Y	fuer das Loeschen der rechts vom Cursor befindlichen Zeichen
CONTROL-V	fuer das Einfuegen einzelner Zeichen
CONTROL-N	fuer das Einfuegen von Zeilen
CONTROL-W	fuer das Abschliessen bzw. Verlassen des Programmeditors.

Als erstes Programm soll nachfolgend das aus dem Abschnitt 3.7.2. bekannte Beispiel der "Kunden mit offenen Rechnungen" als Befehlsdatei gespeichert werden. Schreiben sie

```
MODIFY COMMAND prog1
ERASE
USE auftrag
SELECT SECONDARY
USE kunden
SELECT PRIMARY
JOIN TO kundrech FOR (P.kunr=S.kunr .AND. .NOT. beza .AND. ;
betrag > 0) FIELD vorname,name,plz,ort,strasse,betrag,datum
USE kundrech
```

```
? '      NAME            ORT          STRASSE          BETR;
AG  DATUM'
?
LIST OFF
RETURN
```

Die Liste der Kunden soll mit Spaltenueberschriften versehen werden. Dies wird einfach durch die Funktion "?" - wie oben ausgefuehrt - realisiert. Es ist zu beachten, dass der Text in Hochkommas einzuschliessen ist.

Das Fragezeichen in der naechsten Zeile veranlasst, dass zwischen Ueberschrift und Listenbeginn eine Leerzeile eingefuegt wird.

Der Zusatz OFF im LIST-Befehl hat die Funktion, die Ausgabe der vom System intern vergebenen Datensatznummern zu unterdruecken.

Nachdem Sie mit RETURN die Befehlsfolge beendeten und den Programmeditor durch Betaetigen der <CONTROL-W>-Tastenkombination verlassen haben, wird das Programm auf der Diskette gespeichert (Gleichzeitig wird der Bildschirm geloescht und der Doppelpunkt zeigt Ihnen die Bereitschaft von REDABAS an, weitere Befehle zu empfangen.)

Sie koennen das Programm "prog1" nun mit dem Befehl

DO <befehlsdatei>

wieder aufrufen und abarbeiten. Dies wird in Bild 53 demonstriert.

```
:DO prog1

     NAME            ORT        STRASSE       BETRAG   DATUM

Peter  Lolle     8122 RADEBEUL Goethestr.13     38.38   02.02.85
Gisela Schmidt   8223 THARANDT Bahnhofstr.43   284.81   27.02.85
August Meier     8122 RADEBEUL Uferstr.12     1420.00
Inge   Neumann   8222 RABENAU  Gartenstr. 4    284.81
Karl   Fleischer 8212 FREITAL  Am See 83      1420.00   17.02.85
```

Bild 53 Aufruf und Abarbeiten von "prog1"

Es ist jederzeit moeglich , "prog1" aufzurufen und abzuarbeiten, unabhaengig davon, ob die Dateien "auftrag" oder "kunden" mittlerweile anderweitig bearbeitet wurden.

Will man Befehle veraendern, erfolgt dies mit

MODIFY COMMAND prog1

Das Programm wird in den Arbeitsspeicher geladen und erscheint zur Bearbeitung auf dem Bildschirm. Nach erfolgter Aenderung wird das neue Programm auf einen freien Platz auf der Diskette gespeichert. Die alte Version das Programms (jedoch stets nur die letzte Version) ist als Backup-Datei unter dem Dateinamen

"prog1.BAK" weiterhin verfuegbar. Damit ist sie fuer REDABAS nicht mehr im unmittelbaren Zugriff, koennte aber bei Bedarf in eine ".PRG"-Datei umgewandelt und damit reaktiviert werden. (Dazu dient der Befehl RENAME.)

In einem zweiten Beispiel soll ein sogenannter Kundenbeleg ausgegeben werden, der ausgewaehlte Informationen ueber einen Kunden in Form eines Informationsblattes enthaelt. Diese Informationen sollen Name, Vorname, Kunden-Nr., Adresse, Datum des letzten Auftrages und zu zahlender Betrag sein.

Bevor das entsprechende Programm geschrieben wird, noch die folgenden Informationen.

Mit dem Befehl

@ <koordinaten>

wird es Ihnen ermoeglicht, Ihre Daten auf dem Bildschirm oder auf der Druckliste exakt zu positionieren. Unter "koordinaten" sind konkret Zeile und Spalte zu verstehen, bei der eine zu schreibende Information beginnen soll. Zeile und Spalte sind also Zahlenwerte, die Sie entsprechend Ihrer Bildschirmgroesse oder Ihrer Formularbreite (fuer das Drucken) angeben, d.h. Sie koennen jede moegliche Zeichenposition ansprechen. Die Ausgabe eines Zeichens erfolgt unmittelbar hinter bzw. unter der angegebenen Zahl. Mit

@ 0,0

begaenne also Ihre Information in der 1. Spalte der 1. Zeile. (Das heisst Spalten und Zeilen werden mit 0 beginnend numeriert.)

Sie koennen den Befehl @ noch erweitern fuer die Ausgabe von Text oder Daten. Er lautet dann

@ <koordinaten> SAY <ausdruck>

Unter <ausdruck> werden Text, Namen von Datenfeldern, oder Speichervariablen oder auch Formeln verstanden.

Der Befehl @ kann gut fuer das Formatieren von Druckformularen genutzt werden. Zum einen, weil man Druckpositionen sehr praezis festlegen kann. Zum anderen deshalb, weil man mit diesem Befehl Informationen fuer den Druck markieren kann. Das heisst man kann gezielt nur die Informationen ausdrucken, die mit "@" erzeugt werden und andere Informationen vom Druck ausklammern. Am Anfang des Programmteiles wird dazu der Befehl

SET FORMAT TO PRINT

gegeben, der bedeutet: Ausgabe der formatierten Zeilen auf den Drucker, d.h. diese Zeilen werden nur auf dem Drucker und nicht auf dem Bildschirm ausgegeben.

Soll die Ausgabe der Informationen wieder auf den Bildschirm gelegt werden, schreibt man

SET FORMAT TO SCREEN

Auf diese eise kann man unkompliziert vorgedruckte Formulare, z.B. fuer Rechnungen erstellen.

Zwei weitere Befehle sollen nachfolgend vorgestellt werden:

SET TALK OFF und
SET TALK ON

Sie ermoeglichen es, die Ausgabe der Systemnachrichten zu unterdruecken (OFF) und wieder "einzuschalten" (ON).

Um ein Programm auch waehrend des Programmablaufs noch steuern zu koennen, bietet REDABAS die Moeglichkeit, ueber geeignete Befehle entsprechende Daten einzugeben. Gibt man einen derartigen Befehl an, haelt das Programm an und wartet auf eine Eingabe ueber die Tastatur. Ist diese Eingabe erfolgt, wird die Verarbeitung automatisch fortgesetzt. Einer dieser Befehle lautet

ACCEPT '<text>' TO <speichervariable>

<text> ist eine wahlfreie beliebige alphanumerische Information, die, von einem Doppelpunkt gefolgt auf dem Bildschirm erscheint und als Erlaeuterung fuer die nachfolgende Eingabe gedacht ist, die in <speichervariable> geschrieben werden soll.

Doch nun zurueck zu unserem zweiten Beispiel, dem gewuenschten Ausdruck eines Kundenbeleges, wie ihn Bild 54 zeigt.

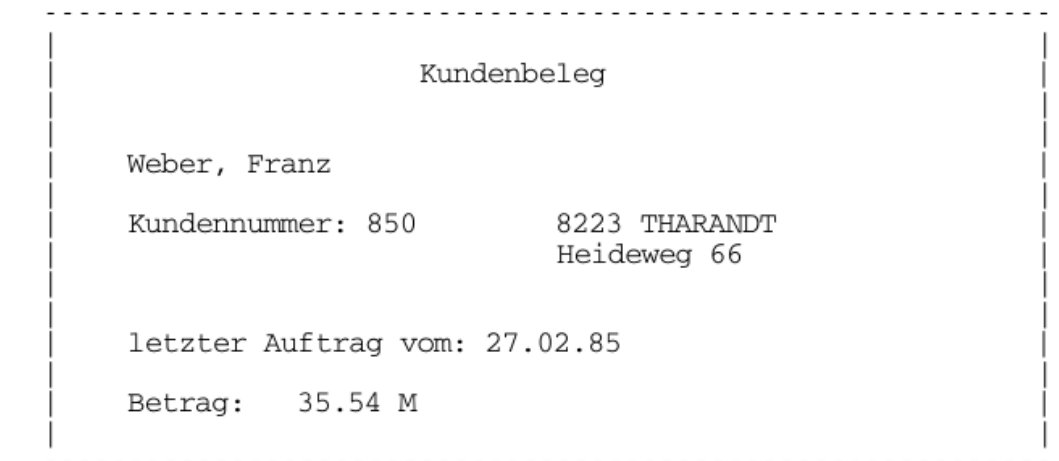

Bild 54 Zusammenstellung wesentlicher Informationen ueber einen Kunden

Das Programm soll "prog2" heissen und Sie schreiben

MODIFY COMMAND prog2

Sie befinden sich jetzt wieder im Programmeditor und koennen in Ihr Programm Erlaeuterungen einfuegen. Dazu dient der Befehl

* <text>

Steht am Anfang der Zeile ein "*", so wird der dem Stern

folgende Text von REDABAS als Kommentar verstanden und bei der Programmausfuehrung uebersprungen. In unserem Beispiel wollen wir zur Kenntlichmachung folgenden Kommentar einfuegen:

```
* Programm Kundenbeleg
```

Danach wird die Ausgabe der Systemnachrichten unterdrueckt mit

```
SET TALK OFF
```

Das eigentliche Programm beginnt wieder mit der Bildschirmloeschung. Danach wird die Ausgabe auf den Drucker zugewiesen und die Ueberschrift auf die 2. Zeile in die 23. Spalte positioniert:

```
ERASE
SET FORMAT TO PRINT
@ 1,22 SAY 'Kundenbeleg'
```

Die Informationen, die wir fuer den Kundenbeleg brauchen, muessen aus zwei unterschiedlichen Dateien entnommen werden. In der Datei "kunden" sprechen wir die Datenfelder "name", "vorname", "plz", "ort", und "strasse" an, in der Datei "auftrag" die Felder "betrag" und "datum". Die Kundennummer ist im Feld "kunr" in beiden Dateien vorhanden und stellt die logische Verbindung zwischen ihnen her. Im folgenden wird die Datei "kunden" als primaere Datei und die nach Kundennummern ueber die Indexdatei "ikunrauf" sortierte Datei "auftrag" als sekundaere Datei eroeffnet. (Diese Indexdatei wurde zur Erhoehung der Verarbeitungsgeschwindigkeit eingefuehrt.)

```
USE kunden
SELECT SECONDARY
USE auftrag INDEX ikunrauf
```

Der Name des Kunden ist bekannt. Um die Anschrift zu finden, muss die Primaerdatei "kunden" durchsucht werden. Ist von vorn herein die Eindeutigkeit des Namens gegeben, genuegt es, nur diesen einzugeben. Ist anzunehmen, dass mehrere Kunden gleichen Namens existieren, ist es zweckmaessig, den Vornamen mit einzugeben.

Die Eingabe des Namens und gegebenenfalls des Vornamens wird auf dem Bildschirm durch eine entsprechende Aufforderung eingeleitet.

Der Name wird in die Speichervariable "Sname" und der Vorname in die Speichervariable "Svorname" gebracht und durch den LOCATE-Befehl mit den entsprechenden Feldern der Datei "kunden" verglichen. Dabei wird der Substring-Operator angewendet, um auch unvollstaendig eingegebene Namen aufzufinden.

```
SELECT PRIMARY
ACCEPT 'Bitte Namen eingeben' TO Sname
ACCEPT 'Bitte Vornamen eingeben' TO Svorname
LOCATE FOR (Sname$name .AND. Svorname$vorname)
```

Stimmen die Namen ueberein, werden aus der Primaerdatei "kunden" die erforderlichen Angaben uebernommen, und sie

koennen mit entsprechender Formatierung durch die folgenden Befehle ausgedruckt werden:

```
@ 5,5 SAY TRIM(name)+','+vorname
@ 7,5 SAY 'Kundennummer:'+' '+kunr
@ 7,30 SAY plz+' '+ort
@ 8,30 SAY strasse
```

Sie lernen hier erstmalig die Funktion

TRIM (<zeichenreihen-ausdruck>)

kennen, die Leerstellen am Ende einer Zeichenreihe entfernt, falls diese vorhanden sind.

In unserem Falle heisst dass, der Name wird auf seine tatsaechliche Laenge verkuerzt, und der Vorname kann ohne unnoetige Leerstellen angeschlossen werden.

Weiterhin sehen Sie, dass man innerhalb einer SAY-Anweisung mehrere Datenfelder oder Zeichen bzw. Texte mittels "+"-Zeichen miteinander verknuepft kann, vorausgesetzt, alle Datenfelder einer Befehlszeile sind alphanumerisch. (In der Druckzeile koennen aber numerische und alphanumerische Felder durch mehrere SAY-Befehle aneinandergereiht erscheinen.)

Die restlichen Informationen fuer den Kundenbeleg muessen aus der Sekundaerdatei entnommen werden. Gemeinsames Identifikationsmerkmal beider Dateien ist die Kundennummer, die fuer die Suche mit FIND in eine Speichervariable gebracht wird (vgl. Abschnitt 3.4.)

```
STORE kunr TO Skunde
SELECT SECONDARY
FIND &Skunde
```

Beim Ausdrucken der restlichen Informationen ist wieder zu beachten, dass numerische und alphanumerische Informationen nicht in einem Druckbefehl stehen duerfen. Man kann dies fuer das Drucken der Zeile "Betrag..." umgehen, indem man zwei Zeilen schriebe, z.B.

```
@ 14,5 SAY 'Betrag:          M'
@ 14,21 betrag
```

Man kann dies aber auch eleganter loesen, indem die String-Funktion angewendet wird:

STR (<numerischer ausdruck>,<laenge>,[<dezimalstellen>])

Mit dieser Funktion wird ein numerischer Ausdruck in eine alphanumerische Zeichenkette umgewandelt.
"numerischer ausdruck" kann eine Zahl, ein Speichername oder ein Feldname sein. Die Laenge des umzuwandelnden Ausdrucks muss angegeben werden, die Angabe der Dezimalstellen ist wahlfrei. Es ist allerdings zu beachten, dass ein mit der String-Funktion umgewandelter numerischer Ausdruck die typischen numerischen Eigenschaften verliert und nicht mehr in arithmetischen Operationen weiterverwendet werden kann.

Wir schreiben also, um die beiden letzten Zeilen des Kundenbeleges zu erstellen:

```
@ 12,5 SAY 'letzter Auftrag vom:' + ' ' + datum
@ 14,5 SAY 'Betrag:' + STR(betrag,8,2) + ' ' + 'M'
```

Anschliessend werden mit dem Befehl

CLEAR

alle Dateien geschlossen und alle temporaeren Speichervariablen geloescht.

SET FORMAT TO SCREEN

legt die Ausgabe der formatierten Daten wieder auf den Bildschirm zurueck und muss in jedem Fall gesetzt werden. Sonst erschienen eventuell in einem spaeterem Programm wichtige Informationen auf dem Drucker anstatt auf dem Bildschirm. Fuer REDABAS gilt allgemein, dass saemtliche Befehle auch ueber das Ende eines Programms hinaus gelten und zwar so lange, bis sie durch Befehle, die deren Wirkung aufheben, ersetzt werden oder bis REDABAS beendet wird. Das gilt ebenfalls fuer das Ansprechen von Dateien. Eine Datei bleibt auch ueber das Ende eines Programms hinaus so lange eroeffnet, bis sie mit dem Eroeffnen einer neuen Datei automatisch geschlossen wird.

In unserem Beispielprogramm wird die Ausgabe der Systemnachrichten wieder zugeschaltet und mit RETURN in die Kommandoebene von REDABAS zurueckgekehrt. Damit sind alle Befehle fuer "prog2" eingegeben, und mit <CONTROL-W> wird die Erstellung des Programms abgeschlossen.

Bild 55 zeigt noch einmal das Programm. Es sei hier darauf verwiesen, dass diese Version des Ausdrucks von Kundenbelegen sehr vereinfacht ist und dass beispielsweise bei falscher Eingabe des Namens das Ergebnis nicht korrekt ist. (vgl. hierzu das Programm "progkun2", Bild 58)

```
* Programm Kundenbeleg
SET TALK OFF
ERASE
SET FORMAT TO PRINT
@ 1,22 SAY 'Kundenbeleg'
USE kunden
SELECT SECONDARY
USE auftrag INDEX ikunrauf
SELECT PRIMARY
ACCEPT ' Bitte  Namen  eingeben' TO Sname
ACCEPT 'Bitte Vornamen eingeben' TO Svorname
LOCATE FOR (Sname¤name .AND. Svorname¤vorname)
@ 5,5 SAY TRIM(name)+', '+vorname
@ 7,5 SAY 'Kundennummer:'+' '+kunr
@ 7,30 SAY plz+' '+ort
@ 8,30 SAY strasse
STORE kunr TO Skunde
SELECT SECONDARY
FIND &Skunde
@ 12,5 SAY 'letzter Auftrag vom:'+' '+datum
@ 14,5 SAY 'Betrag:'+STR(betrag,8,2)+' '+'M'
CLEAR
SET FORMAT TO SCREEN
SET TALK ON
RETURN
```

Bild 55 "prog2" - Programm zur Erstellung eines Kundenbeleges

Wollen Sie nun Kundenbelege ausdrucken, rufen Sie das Programm mit

DO prog2

auf. Sie werden um die Eingabe des Namens gebeten und erhalten nach kurzer Zeit den entsprechenden Beleg ausgedruckt.

4.2. Programmschleifen

Im Abschnitt 3.7.1. lernten Sie im Rahmen des gleichzeitigen Bearbeitens von zwei Dateien ein Programm kennen, mit dem die Kosten der Teile eines Auftrages und dessen Gesamtkosten ermittelt wurden (vgl. Bild 47). Schon dort wurde erkannt, dass die angewandte Methode, fuer jeden Datensatz die gleiche Befehlsfolge ablaufen zu lassen, recht uneffektiv ist.

Fuer derartiges wiederholtes Abarbeiten bietet sich der Einbau einer Programmschleife an, wodurch eine abgegrenzte Befehlsfolge beliebig oft durchlaufen werden kann.

Nachfolgend soll demonstriert werden, wie durch den Einbau einer Programmschleife das mehrfache Eingeben gleicher Befehlsfolgen eingespart wird.

Sie sehen dies im Programm "rechnung" im Bild 56 dargestellt.

```
* Kundenrechnung erstellen
ERASE
SELECT PRIMARY
USE aufpos
SELECT SECONDARY
USE bestand INDEX iteilbes
SELECT PRIMARY
DO WHILE .NOT. EOF
   STORE teilnr TO Steil
   SELECT SECONDARY
   FIND &Steil
   SELECT PRIMARY
   REPLACE P.betrag WITH S.preis*P.anz
   SKIP
ENDDO
USE aufpos INDEX inaufpos
TOTAL ON aufnr TO rech
USE rech
REPLACE ALL datum WITH date()
USE auftrag INDEX iaufnrau
UPDATE FROM rech ON aufnr REPLACE betrag,datum
CLEAR
RETURN
```

Bild 56 Programm "rechnung", mit Programmschleife realisiert

Wir moechten einige Erlaeuterungen zu den noch unbekannten Befehlen geben.

Bewusst wurde am Programmanfang die Datei "aufpos" mit SELECT PRIMARY eroeffnet. Denn REDABAS-Befehle wirken ueber das Ende eines Programms hinaus (wie Sie bereits im letzten Abschnitt erfuhren), und es koennte sein, dass ein frueher (in einem anderen Programm) gegebener Befehl noch wirksam waere, der die Datei "aufpos" als sekundaere Datei reserviert hat und somit einen Programmabsturz veranlassen koennte.

Der Befehl fuer das Eroeffnen einer Programmschleife lautet

DO WHILE <bedingung>

Die diesem Befehl folgenden Befehle werden solange durchlaufen, wie die <bedingung> erfuellt ist.
Innerhalb der Programmschleife werden die Befehle ausgefuehrt, bis mit

ENDDO

das Ende der DO WHILE-Schleife markiert ist. Danach wird zum DO WHILE-Befehl zurueckgesprungen und die <bedingung> erneut ausgewertet.

Ist die <bedingung> nicht mehr erfuellt, wird die Schleife verlassen und das Programm mit dem auf ENDDO folgenden Befehl fortgesetzt.

Im vorliegenden Beispiel lautet der Beginn der Programmschleife

DO WHILE .NOT. EOF

Das heisst, die Schleife soll solange ausgefuehrt werden, bis das Ende der Datei (hier das der Datei "aufpos") erreicht wird. Ist dies erreicht, wird mit dem Befehl

USE aufpos INDEX inaufpos

usw. fortgefahren.

Die weiteren Befehle zur Erstellung der Rechnung werden hier nicht naeher erlaeutert, da sie schon aus dem Abschnitt 3.7.1. bekannt sind.

Bemerkenswert ist noch, dass es wegen der besseren Uebersichtlichkeit und Lesbarkeit eines Programms zweckmaessig ist, die Befehle, die zur DO WHILE-Schleife gehoeren, etwas einzuruecken.

Die Anwendung einer Programmschleife soll noch an einem zweiten Beispiel veranschaulicht werden. Ausgangspunkt dafuer ist die Erstellung des Kundenbeleges, wie Sie ihn im vorigen Abschnitt kennenlernten. Wollen Sie eine Anzahl von Kundenbelegen drucken, muessen Sie jedesmal das in Bild 55 dargestellte "prog2" mit "DO prog2" aufrufen. Und jedesmal muessen zwei Dateien eroeffnet werden.

Auch hier bietet sich der Einbau einer Programmschleife an, die je nach der Zahl der gewuenschten Kundenbelege abgearbeitet wird .

Sie sehen das entsprechende ausgebildete Programm im Bild 57.

```
* Programm Kundenbeleg
SET TALK OFF
SELECT PRIMARY
USE kunden
SELECT SECONDARY
USE auftrag INDEX ikunrauf
STORE 'J' TO weiter
DO WHILE !(weiter) <> 'N'
   ERASE
   SET FORMAT TO PRINT
   @ 1,25 SAY 'Kundenbeleg'
   ACCEPT 'Name des Kunden' TO Sname
   ACCEPT 'Vorname des Kunden' TO Svorname
   SELECT PRIMARY
   LOCATE FOR (Sname¤name .AND. Svorname¤vorname)
   @ 5,5 SAY TRIM(vorname)+', '+name
   @ 5,30 SAY 'Kunden-Nr'+' '+kunr
   @ 7,5 SAY plz+' '+ort
   @ 8,5 SAY strasse
   STORE kunr TO Skunde
   SELECT SECONDARY
   FIND &Skunde
   @ 12,5 SAY 'letzter Auftrag vom:'+datum
   @ 13,5 SAY 'Betrag:'+STR(betrag,8,2)
   ? 'Fuer weitere Kundenbelege RETURN, sonst N ;
   druecken!'
   WAIT TO weiter
ENDDO
SET FORMAT TO SCREEN
CLEAR
SET TALK ON
RETURN
```

Bild 57 Programm "progkun1" zur Erstellung von Kundenbelegen, mit Programmschleife realisiert

Wieder sollen einige unbekannte Befehle bzw. Funktionen erlaeutert werden.

Das Durchlaufen der DO WHILE-Schleife wird hier vom Inhalt einer Speichervariablen abhaengig gemacht. Um die Schleife ueberhaupt durchlaufen zu koennen, muss die Bedingung so gesetzt sein, dass sie anfangs auf jeden Fall erfuellt ist.

Es wird vor der ersten Abfrage ein "J" (JA) in den Speicher namens "weiter" gebracht:

```
STORE 'J' TO weiter
```

Spaeter kann der Speicherinhalt auf "N" (NEIN) gesetzt werden, die darauffolgende Abfrage wird dann zum Verlassen der Programmschleife fuehren. Mit

```
DO WHILE !(weiter) <> 'N'
```

wird die Programmschleife eroeffnet. Das bedeutet, die Schleife soll solange durchlaufen werden, wie der Inhalt der

Speichervariablen "weiter" ungleich "N" ist. Das Ausrufezeichen nach DO WHILE ist Kennzeichen der sogenannten Grossbuchstaben-Funktion:

!(<zeichenreihen-ausdruck>)

Diese Funktion wandelt alle im <zeichenreihen-ausdruck> vorkommenden Kleinbuchstaben in Grossbuchstaben um. Damit ist gesichert, dass auch eingegebene Kleinbuchstaben als Grossbuchstaben gewertet werden und im vorliegenden Fall mit Gross-N verglichen werden.

WAIT TO weiter

Damit stellen Sie die Frage, ob die Programmschleife erneut abzuarbeiten ist oder ob zum Programmende uebergegangen wird. Der Befehl

WAIT TO <speichervariable>

veranlasst eine Pause bei der Abarbeitung eines Programms, um ueber die Tastatur eine Eingabe machen zu koennen. Im vorliegenden Fall wird nach "WAIT TO weiter" bei Eingabe eines "N" oder "n" das Programm zum Ende gefuehrt.

Im Unterschied zu den Befehlen ACCEPT und INPUT, wo ganze Worte eingegeben werden, nimmt WAIT nur eine einzelne Taste als Eingabe an und bietet so Moeglichkeiten fuer schnelle Eingaben. WAIT braucht nicht mit <RETURN> abgeschlossen zu werden.

WAIT kann auch ohne den Zusatz TO <speichervariable> benutzt werden. WAIT veranlasst dann lediglich eine Pause bei der Abarbeitung eines Programms. Nach einem beliebigen Tastendruck wird fortgesetzt. Will man beispielsweise Zwischenergebnisse eines Programms anzeigen lassen, kann man auf diese Weise eine Unterbrechung des Programms veranlassen.

In unserem vorliegenden Beispiel wird die Ausgabe von Kundenbelegen solange fortgesetzt, wie irgend eine Taste ausser "N" gedrueckt wird. Erst mit "N" wird das Programm verlassen.

Bestimmt haben Sie an den beiden gezeigten Beispielen die Vorteile des Programmierens mittels Programmschleife erkannt. Waehrend Ihnen im ersten Beispiel vor allem eine Menge an Eingaben erpart blieb, werden im zweiten Beispiel die wesentlich kuerzeren Antwortzeiten des Systems infolge der nur einmal zu eroeffnenden Dateien bestechen.

Sie haben die Moeglichkeit (und in der Praxis besteht dazu meist auch die Notwendigkeit), die erstellten Programme zu protokollieren

Verlassen Sie dazu REDABAS und lassen Sie die Programme nach Betaetigen der Tastenkombination <CONTROL-P> mit den SCP-Befehlen

TYPE prog2.PRG bzw.

TYPE progkun1.PRG

drucken. Erforderlich ist hier - wie auch bei Verwendung des Textverarbeitungssystems zum Drucken - die vollstaendige Angabe des Dateinamens.

4.3. Programmverzweigungen

REDABAS bietet zwei Moeglichkeiten, Verzweigungen in einem Programm zu realisieren.

Die erste Variante stellt die folgende Befehlsgruppe dar:

```
IF <bedingung>
    :
    Befehle fuer erfuellte Bedingung
    :
ELSE
    :
    Befehle fuer nicht erfuellte Bedingung
    :
ENDIF
```

Die zweite Variante lautet:

```
DO CASE
CASE <bedingung>
     :
     Befehle fuer erfuellte 1. Bedingung
     :
CASE <bedingung>
     :
     Befehle fuer erfuellte n-te Bedingung
     :
ENDCASE
```

Zunaechst soll die erste Moeglichkeit an einem Programm, das Sie bereits im vorigen Abschnitt (s. Bild 57) kennenlernten, demonstriert werden. Das erweiterte Programm, das jetzt "prokun2" heissen soll, sehen Sie im Bild 58 dargestellt.

```
* Programm Kundenbeleg
SET TALK OFF
SELECT PRIMARY
USE kunden
SELECT SECONDARY
USE auftrag INDEX ikunrauf
STORE 'J' TO weiter
DO WHILE !(weiter) <> 'N'
   ERASE
   SET FORMAT TO PRINT
   @ 1,25 SAY 'Kundenbeleg'
   ACCEPT 'Name des Kunden' TO Sname
   ACCEPT 'Vorname des Kunden' TO Svorname
   SELECT PRIMARY
   LOCATE FOR (Sname¤name .AND. Svorname¤vorname)
   @ 5,5 SAY TRIM(vorname)+', '+name
   @ 7,5 SAY 'Kundennummer'+' '+kunr
   @ 7,30 SAY plz+' '+ort
   @ 8,30 SAY strasse
   STORE kunr TO Skunde
   SELECT SECONDARY
   FIND &Skunde
   IF #=0
        SET FORMAT TO SCREEN
        ERASE
        @ 12,5 SAY 'Kunde nicht vorhanden!'
        @ 13,5 SAY '----------------------'
   ELSE
        SET FORMAT TO PRINT
        @ 12,5 SAY 'letzter Auftrag vom:'+datum
        @ 13,5 SAY 'Betrag:'+STR(betrag,8,2)
   ENDIF
   ? 'Fuer weitere Kundenbelege RETURN, sonst N druecken|'
   WAIT TO weiter
ENDDO
SET FORMAT TO SCREEN
CLEAR
SET TALK ON
RETURN
```

Bild 58 Programm "progkun2" zur Erstellung von Kundenbelegen- mit Programmverzweigung

Vergleichen Sie beide Programme ("progkun1" und "progkun2"), erkennen Sie, dass sie sich bis zum Befehl "FIND &Skunde" keineswegs unterscheiden. "progkun2" beruecksichtigt jedoch den Fall, dass der mit dem FIND-Befehl gesuchte Datensatz nicht vorhanden ist. Ein erfolgloser FIND-Befehl setzt den Datensatzzeiger auf die Satznummer 0. In Abhaengigkeit davon, ob die Datensatznummer 0 ist oder nicht (ob also der Datensatz existiert oder nicht), wird verzweigt.

IF #=0 : Auf dem Bildschirm erscheint die Nachricht
"Kunde nicht vorhanden!"
(Mit # wurde bekanntlich die Satznummer bezeichnet.)

ELSE : Sonst, d.h. die Datensatznummer ist ungleich 0, wird weiterverfahren wie bereits im Programm "progkun1" realisiert, d.h. der Kundenbeleg wird vollstaendig ausgedruckt.

Damit erkennt man die Wirkungsweise des IF-Befehls: Ist die IF-<bedingung> erfuellt, werden die unmittelbar folgenden Befehle ausgefuehrt, andernfalls kommen die Befehle zur Ausfuehrung, die durch ELSE eingeleitet werden. Enthaelt ein IF-Befehl keinen ELSE-Teil, werden bei Nichterfuellung der IF-Bedingung alle Befehle bis zum naechsten ENDIF-Befehl uebergangen. Ein mit IF eingeleiteter Befehlskomplex muss stets mit dem Befehl ENDIF abgeschlossen sein.

Es ist zweckmaessig, anstelle grosser und komplexer Anwendungsprogramme mehrere kleinere zu entwickeln, diese miteinander zu verknuepfen und nur das jeweils gewuenschte aufzurufen.

Eine vorteilhafte Methode ist die Anwendung der Menuetechnik fuer derartig strukturierte Programme. Auf dem Bildschirm werden in Form einer Uebersicht alle vorhandenen Unterprogramme bzw. deren Funktionen angezeigt. Der Nutzer waehlt ueber die Tastatur das von ihm gewuenschte Programm aus, REDABAS arbeitet es ab und kehrt danach zur Programmuebersicht zurueck, der Nutzer kann erneut auswaehlen.

Im folgenden demonstrieren wir die zweite Verzweigungsvariante mit der DO CASE-Befehlsgruppe anhand eines Menueprogramms, das wiederum auf die Ihnen vertraute Problematik eines Reparaturbetriebes orientiert und die bekannten Dateien auswertet. Folgende Funktionen sollen realisiert werden

- das Erfassen neuer Kundenauftraege
- das Erstellen von Rechnungen
- das Aktualisieren des Lagerbestandes
- das Anzeigen des Ersatzteilbedarfs
- das Ausdrucken eines Kundenbeleges.

Bild 59 zeigt die Uebersicht, die dem Nutzer zur Auswahl auf dem Bildschirm angezeigt werden soll. Bild 60 gibt das Menueprogramm wieder.

Neben dem Aufruf eines Programms mit dem DO-Befehl, kann der Programmname auch als Parameter beim REDABAS-Start mitgeteilt werden. Durch Eintippen von

```
REDABAS menue1
```

ruft REDABAS nach Ausgabe der Startmeldung selbstaendig das genannte Programm auf.

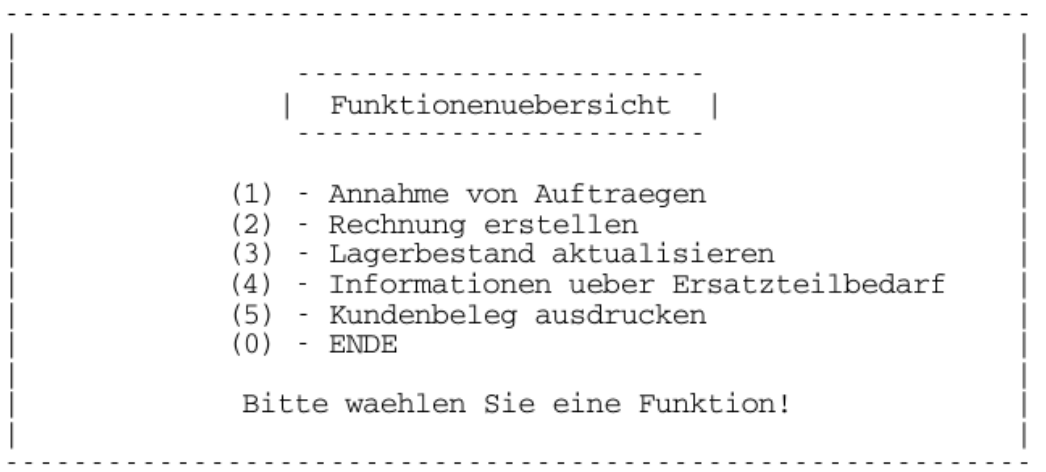

Bild 59: Menueauswahl

```
* Menueprogramm
STORE T TO test
DO WHILE test
   ERASE
   @ 1,20 SAY '------------------------'
   @ 2,19 SAY '|  Funktionenuebersicht  |'
   @ 3,20 SAY '------------------------'
   @ 5,15 SAY '(1) - Annahme von Auftraegen'
   @ 6,15 SAY '(2) - Rechnung erstellen'
   @ 7,15 SAY '(3) - Lagerbestand aktualisieren'
   @ 8,15 SAY '(4) - Informationen ueber Ersatzteil;
   bedarf'
   @ 9,15 SAY '(5) - Kundenbeleg ausdrucken'
   @ 10,15 SAY '(0) - ENDE'
   @ 12,15 SAY 'Bitte waehlen Sie eine Funktion!'
   @ 13,0
   WAIT TO auswahl
   DO CASE
      CASE auswahl = '1'
        REMARK Unterprogramm "aufnahme" wird noch
        erstellt
        WAIT
      CASE auswahl = '2'
        DO rechnung
      CASE auswahl = '3'
        ERASE
        USE aufpos INDEX iteilpos
        TOTAL ON teilnr TO reorg
        USE reorg
        REPLACE ALL anz WITH anz*(-1),datum WITH date()
        USE bestand INDEX iteilbes
        UPDATE FROM reorg ON teilnr ADD anz REPLACE datum
      CASE auswahl = '4'
        USE bestand INDEX iortbest
        SET HEADING TO Bedarfsliste
        REPORT FORM best31 FOR anz<minbest
      CASE auswahl = '5'
        ERASE
        @ 5,1 SAY 'Bitte erst Drucker einschalten...'
        @ 6,1 SAY 'danach mit ^RETURN_-Taste fortsetzen!'
        @ 8,1
        WAIT
        DO progkun2
      CASE auswahl = '0'
        RETURN
   ENDCASE
ENDDO
```

Bild 60 Menueprogramm "menue1"

Nachfolgend werden unbekannte Passagen des Programms erklaert. Der erste Programmteil realisiert die Anzeige des Menues. Der DO-WHILE-Befehl wird hier wieder in Abhaengigkeit vom Inhalt der Speichervariablen "test" gesetzt. Es wird jetzt kein alphanumerischer Wert in die Speichervariable gebracht (der in Hochkommas gesetzt werden muesste), sondern ein in REDABAS fest definierter Wert. Logische Werte koennen T (fuer true = wahr) und J (fuer ja) oder deren Gegenteil F (fuer false = falsch) und N (fuer nein) sein und stehen ohne Hochkommas.

Die Programmschleife wird solange durchlaufen, wie die Speichervariable "test" den logischen Wert "T" beinhaltet. Innerhalb des Programms wird der Wert nicht veraendert, das Verlassen der Programmschleife wird ueber die Programmverzweigung und den folgenden RETURN-Befehl realisiert. Will man den logischen Wert innerhalb eines Programms nicht veraendern, kann man den Befehl auch vereinfachen und schreibt

```
DO WHILE T
```

Nachdem die entsprechenden Ausschriften auf dem Bildschirm positioniert wurden, kann nach

```
WAIT TO auswahl
```

durch Eingabe einer entsprechenden Ziffer in eines der zur Auswahl stehenden Unterprogramme verzweigt werden. 6 Faelle werden unterschieden.

Diese zweite Moeglichkeit der Programmverzweigung, die man auch als Fallauswahl bezeichnen kann, wird mit DO CASE eingeleitet und mit ENDCASE abgeschlossen. Die Anzahl der dazwischenliegenden CASE-Klauseln ist nicht begrenzt. Ist eine der CASE-Bedingungen erfuellt, werden die darauffolgenden uebersprungen.

Im Beispiel wird in Abhaengigkeit von der jeweils gedrueckten Zifferntaste in ein Programm verzweigt:

Ziffer 1:
Es soll das spaeter in Bild 61 dargestellte Programm "aufnahme" abgearbeitet werden, das dem Erfassen neuer Auftraege dient. Dieses sogenannte "Online-Programm" existiert zur Zeit noch nicht (Sie lernen es im folgenden Abschnitt kennen). Deshalb sollte man in dem Fall keine Anweisung, wie z.B. "DO aufnahme" schreiben. Das (nicht vorhandene) Unterprogramm wuerde gesucht, nicht aufgefunden und das gesamte Menueprogramm wuerde mit einer Fehlernachricht abgebrochen.

Mit dem Befehl

```
REMARK <text>
```

ist es moeglich , einen beliebiger <text> als Kommentar auszugeben

Als Hinweis schreiben wir hier:
REMARK Unterprogramm "aufnahme" wird noch erstellt!

Ziffer 2:
Das aus dem Abschnitt 4.2. und Bild 56 bekannte Programm "rechnung" wird zum Zweck der Erstellung von Rechnungen zur Verarbeitung aufgerufen.

Ziffer 3:
Der Lagerbestand soll aktualisiert werden. Dazu werden nach Loeschen des Bildschirms sowie Eroeffnen der nach Teilnummern indizierten Datei "aufpos" die Datensaetze mit jeweils gleicher Teilnummer zusammengefasst. In der Datei "reorg", die zuvor mit dem CREATE-Befehl (1. Feld: teilnr, c ,5; 2. Feld: anz, n, 4; 3. Feld: datum, c, 8) zu erstellen ist, wird der Inhalt der Felder "anz" mit (-1) multipliziert, so dass diese dann ueber den UPDATE-Befehl mit dem Zusatz ADD in der Datei "bestand" aktualisiert werden koennen. Gleichfalls wird das Datum aktualisiert. Die komplette Befehlsfolge ist hier (im Gegensatz zu Fall 2 und 5) direkt im Menueprogramm angeordnet.

Ziffer 4:
In diesem Fall soll eine Liste ueber den Ersatzteilbedarf ausgegeben werden, wie sie bereits im Abschnitt 3.6. im Bild 44 erstellt wurde.

Ziffer 5:
Es wird das Programm "progkun2" aufgerufen. Zuvor wird eine Aufforderung zum Einschalten des Druckers ausgegeben.

Ziffer 0:
Das Menueprogramm wird beendet, da hier nur der Befehl RETURN zur Ausfuehrung kommt. Das Druecken der Zifferntaste "0" ist die einzige Moeglichkeit, das Menueprogramm zu verlassen.

Das Programm "menue1" kann in dieser Weise abgearbeitet werden. Momentan besteht nur die einzige Einschraenkung bei Druecken von Zifferntaste 1. Druecken Sie diese Taste, erscheint der entsprechend programmierte Hinweis und das Menue wird wieder angeboten.

4.4. Programmierte Dateneingabe

Zu einem komfortablen Datenbankbetriebssystem gehoeren auch Moeglichkeiten der direkten Erfassung und sofortigen Verarbeitung von Daten in einem laufenden Progamm.

Befehle fuer die Erfassung von Daten ueber die Tastatur, die Sie innerhalb dieser Programmtechnischen Beschreibung schon kennenlernten, waren beispielsweise ACCEPT und APPEND. So wird bei Anwendung von APPEND automatisch eine Erfassungsmaske zur Verfuegung gestellt, die aber stets alle Datenfelder in derselben Reihenfolge, wie sie definiert wurden, enthaelt. Oft wird dies fuer eine gezielte Eingabe bzw. Aenderung von einzelnen ausgewaehlten Datenfeldern uneffektiv sein.

Im Abschnitt 4.1. lernten Sie den Befehl @ kennen. @ kann um die sogenannte GET-Klausel erweitert werden und ermoeglicht in dieser Form die direkte Dateneingabe auf folgende Weise:

GET <variable>

bewirkt die Anzeige des Inhalts des unter <variable> spezifizierten Datenfeldes oder einer Speichervariablen der vom Nutzer mit dem nachfolgenden Befehl

READ

ueberschrieben werden kann.

Eine wahlfreie Klausel

PICTURE <maske>

dient der Formatierung und auch der Plausibilitaetspruefung der Eingabe. Es koennen sowohl numerische als auch alphanumerische und logische Daten eingegeben werden. (Der Cursor kann bei der Eingabe ebenso wie beim gewoehnlichen Editieren vor- und zurueckbewegt werden.)

Im folgenden Beispiel sehen Sie das eben Gesagte demonstriert:

```
------------------------------------------------------------
|                                                          |
| * Erfassen neuer Auftraege                               |
| ERASE                                                    |
| USE auftrag                                              |
| APPEND BLANK                                             |
| @ 2,5 SAY 'Auftrags-Nr.' GET aufnr                       |
| @ 3,5 SAY 'Kunden-Nr.' GET kunr PICTURE 'XXX'            |
| @ 4,5 SAY 'Datum ' GET datum PICTURE '99.99.99'          |
| READ                                                     |
| RETURN                                                   |
|                                                          |
------------------------------------------------------------
```

Bild 61 Programm "aufnahme"

Das Programm "aufnahme", ein Unterprogramm des Menueprogramms (s. Bild 60), hat das Erfassen neuer Kundenauftraege zum Inhalt. Diese Auftraege sollen direkt (online) ueber Tastatureingabe mit Auftragsnummer, Kundennummer und aktuellem Datum erfasst werden. (Auftraege sind, wie Sie im Abschnitt 3.5. erfuhren, in der Datei "auftrag" gespeichert.)

Eine entsprechende Erfassungsmaske wird wie folgt erstellt:

Nach dem Loeschen des Bildschirms und dem Eroeffnen der Datei "auftrag" wird mit dem Befehl

APPEND BLANK

an diese Datei ein leerer Datensatz angehaengt. Der Bildschirm wird nun mit mehreren @-Befehlen gefuellt. Die PICTURE-Klausel gibt in der <maske> eine Zeichenfolge an und definiert damit die gueltige Art der Dateneingabe. XXX bedeutet also: Eingabe von max. drei beliebigen Zeichen, waehrend 99.99.99 nur die Eingabe von Ziffern in dieser Form (Datum) erlaubt.

Mit dem Befehl

READ

koennen nun die Daten fuer die drei Felder eingegeben werden, wobei sich der Cursor mittels der <CONTROL>-Tasten frei in diesen Feldern bewegen laesst, vorausgesetzt der Seitenmodus ist aktiv. Durch

RETURN

wird die Befehlsfolge abgeschlossen.

Sie lernten mit diesem Befehl eine Moeglichkeit der Online-Verarbeitung kennen, naemlich die direkte Erfassung von Daten. Auf diese Weise koennen Sie das Schema fuer die Erfassung und Aktualisierung Ihrer Datenbestaende selbst programmieren und trotzdem den vollen REDABAS-Service bzgl. Dateiaktualisierung und Cursorsteuerung nutzen.

Bezueglich des Programms "aufnahme" verbleibt uns nunmehr nur noch die Aufgabe des Einbindens in das Menueprogramm. Dazu sind lediglich die Befehle REMARK...und WAIT nach "CASE auswahl = '1'" durch den Befehl "DO aufnahme" zu ersetzen (vgl. Bild 62).

```
* Menueprogramm
STORE T TO test
DO WHILE test
   ERASE
   @ 1,20 SAY '------------------------'
   @ 2,19 SAY '|  Funktionenuebersicht  |'
   @ 3,20 SAY '------------------------'
   @ 5,15 SAY '(1) - Annahme von Auftraegen'
   @ 6,15 SAY '(2) - Rechnung erstellen'
   @ 7,15 SAY '(3) - Lagerbestand aktualisieren'
   @ 8,15 SAY '(4) - Informationen ueber Ersatzteil;
   bedarf'
   @ 9,15 SAY '(5) - Kundenbeleg ausdrucken'
   @ 10,15 SAY '(0) - ENDE'
   @ 12,15 SAY 'Bitte waehlen Sie eine Funktion!'
   @ 13,0
   WAIT TO auswahl
   DO CASE
      CASE auswahl = '1'
        DO aufnahme
      CASE auswahl = '2'
        DO rechnung
      CASE auswahl = '3'
        ERASE
        USE aufpos INDEX iteilpos
        TOTAL ON teilnr TO reorg
        USE reorg
        REPLACE ALL anz WITH anz*(-1),datum WITH date()
        USE bestand INDEX iteilbes
        UPDATE FROM reorg ON teilnr ADD anz REPLACE datum
      CASE auswahl = '4'
        USE bestand INDEX iortbest
        SET HEADING TO Bedarfsliste
        REPORT FORM best31 FOR anz<minbest
      CASE auswahl = '5'
        ERASE
        @ 5,1 SAY 'Bitte erst Drucker einschalten...'
        @ 6,1 SAY 'danach mit <RETURN>-Taste fortsetzen!'
        @ 8,1
        WAIT
        DO progkun2
      CASE antwort = '0'
        RETURN
   ENDCASE
ENDDO
```

Bild 62 "menue2" - komplettes Programm zur Erstellung des Bildschirmmenues

4.5. Zusammenfassung

Der Abschnitt 4. zeigte Ihnen insgesamt neue leistungsfaehige Befehle fuer die Programmierung von Anwendungsprogrammen, die sehr komplex sein koennen. Sie erfuhren, wie Befehlsfolgen in Befehlsdateien zusammengefasst, gespeichert und abgearbeitet werden koennen.

REDABAS ist mit diesen Moeglichkeiten ein sehr leistungsstarkes Datenbankbetriebssystem, das sowohl fuer taegliche Dialogabfragen und -auswertungen als auch fuer sehr komlexe menuegesteuerte Anwendungsprogramme eingesetzt werden kann.

Zusammenfassend kann gesagt werden, dass Sie im Abschnitt 4. die wichtigsten Befehle und Funktionen von REDABAS kennengelernt haben. Die Befehle wurden in der Reihenfolge beschrieben, wie sie in der Praxis angewandt werden. Demonstrationsbeispiele wurden zum besseren Verstaendnis beigefuegt und ausfuehrlich beschrieben, so dass sie von jedem Anwender leicht nachvollzogen werden koennen, und er sehr schnell erste Erfolge bei der Arbeit mit REDABAS erzielen kann. Einschraenkend muss gesagt werden, dass im Rahmen dieses einfuehrenden Teils der Programmtechnischen Beschreibung nicht alle ausfuehrbaren Befehle und Funktionen und deren Varianten vorgefuehrt werden konnten. Ziel der bisherigen Beschreibung war es, dem Anwender so schnell wie moeglich einen Einstieg in das Datenbankbetriebssystem REDABAS zu ermoeglichen, ihn mit dessen wesentlichstem Merkmal, der zugehoerigen Programmiersprache vertraut zu machen und damit auch das immense Leistungsspektrum von REDABAS zu verdeutlichen.

In den folgenden Abschnitten werden Sie eine genaue Beschreibung des gesamten Vorrats an Befehlen, Funktionen, Dateien usw. finden und erhalten damit ein lueckenloses Arbeitsmittel fuer die spaetere Arbeit mit REDABAS.

5. Datenorganisation

5.1. Aufbau der Datenbank

Das relationale Datenbankbetriebssystem REDABAS organisiert die Datenbestaende in Form von Tabellen und kommt damit den Vorstellungen des Anwenders entgegen. Eine zweidimensionale Tabelle wird durch eine Relation realisiert, die im traditionellen Sinn einer Datei entspricht. Die Spalten der Tabelle werden durch die Datenfelder verkoerpert, die Zeilen der Tabelle entsprechen den Datensaetzen.

Teile-Nr.	Bezeichnung	Anzahl	Preis	Lager-ort
00819	Bremsleitung	31	5.80	140
00852	Bolzen	471	0.10	110
14043	Ansaugkruemmer	194	2.45	140
17000	Blinkleuchte, vorn	114	10.00	130

Bild 63 Tabelle eines Lagerbestandes

Die Informationen zum Tabellenaufbau, d.h. Spaltenbezeichnung (Feldname), Feldtyp (alphanumerisch, numerisch oder logisch) und Spaltenbreite (Feldlaenge) hinterlegt REDABAS im Kopf der Datenbankdatei, bevor die Speicherung der Tabellenwerte erfolgt. Bei der Erfassung einer Tabelle speichert REDABAS die Zeilen als Datensaetze sequentiell in der Eingabereihenfolge ab.

REDABAS kann beliebig viele Tabellen als Datenbankdateien speichern. Die interne Darstellung der Feldwerte entspricht dem externen Format, es erfolgt keine Konvertierung.

Der Platzbedarf einer Datenbankdatei auf Diskette (in Byte) errechnet sich aus:

(Summe der Feldlaengen + 1) * Anzahl der Saetze + 521

Zu beachten ist, dass SCP auf der Diskette Mindestbereiche von 2 K Bytes reserviert.

REDABAS verlangt fuer die Datensaetze kein Sortierkriterium. Wuenscht der Nutzer jedoch eine bestimmte Reihenfolge der Datensaetze, so kann er eine Sortierung oder Indizierung des Datenbestandes herbeifuehren (vgl. Abschnitt 3.4. und 8.3.). Bei einer Sortierung (mit dem Befehl SORT) erfolgt eine physische Umordnung der Datensaetze. Bei Dateierweiterungen muss der Nutzer fuer die Einhaltung der Sortierordnung selbst Sorge tragen, indem er die Position fuer die neuen Saetze bestimmt, oder es muss ein neuer Sortiervorgang eingeleitet werden.

Eine zweite Methode zur Erreichung einer Sortierordnung ist die Indizierung (mit dem Befehl INDEX). Bei der Indizierung wird eine zusaetzliche Datei erstellt, die zu jedem Datensatz den Schluesselbegriff und einen Zeiger zum Datensatz enthaelt.

Solche Indexdateien erlauben einen schnellen Zugriff zu einzelnen Datensaetzen. Ausserdem koennen Indexdateien beim Einfuegen, Loeschen und Aendern von Datensaetzen automatisch aktualisiert werden. REDABAS indiziert auf Wunsch beliebige Datenfelder.

5.2. Datentypen und Feldformat

REDABAS unterstuetzt drei verschiedene Datentypen:

Typ "C" - alphanumerisch:
Dieser Typ erlaubt alle druckbaren ASCII-Zeichen wie Buchstaben, Zahlen, Leerzeichen und Sonderzeichen, die mit der Tastatur erzeugt werden koennen. Andere Zeichen, die nicht druckbar sind, aber mit den Steuertasten der Tastatur auch erzeugt werden koennen, sind nicht zugelassen. Befinden sich solche Zeichen in Datenfeldern, koennen unkontrollierte Bildschirm- oder Programmeffekte auftreten.

Typ "N" - numerisch:
Numerische Felder koennen positive und negative Zahlen im Wertebereich von

$$\text{ca. } 1{,}0 \cdot 10^{63} \text{ bis } 1{,}0 \cdot 10^{-63}$$

enthalten. Die Zahlengenauigkeit betraegt max. 10 Stellen. Das heisst, bei der Dateneingabe werden nur 10 Stellen - beginnend bei der ersten von Null verschiedenen Stelle - beruecksichtigt. Das gleiche trifft fuer Rechenoperationen zu, die Stellen ausserhalb des Genauigkeitsbereiches werden abgeschnitten. Numerische Felder koennen ein Vorzeichen und/oder einen Dezimalpunkt enthalten. Bei der Laengenangabe in der Felddefinition sind diese Zeichen mit zu beruecksichtigen.

Typ "L" - logisch:
Logische Felder sind stets ein Zeichen lang und enthalten den Wahrheitswert "wahr" oder "falsch" (engl.: "true" oder "false"). Bei der Eingabe sind folgende Alternativangaben moeglich:

fuer "wahr" : T, t, J, j
fuer "falsch": F, f, N, n

In der Strukturdefinition bei Ausfuehrung des CREATE-Befehls sind fuer alphanumerische und numerische Felder die Feldlaengen anzugeben. REDABAS arbeitet mit festen Satzlaengen, d.h. es wird fuer jeden Datensatz einer Datei auf Diskette ein gleich grosser, durch die Strukturdefinition festgelegter, Platz reserviert. Feldwerte, die nicht die maximale Feldlaenge ausnutzen, werden automatisch durch Leerstellen ergaenzt.

5.3. REDABAS - Dateien

Allgemein versteht man unter einer Datei eine Sammlung von Daten, die auf einem Massenspeicher (z.B. Diskette) aufbewahrt werden. Das Betriebssystem SCP unterstuetzt diese Datenspeicherung und organisiert auf dem Speicher ein Verzeichnis, in dem alle darauf befindlichen Dateien unter einem Namen eingetragen sind. Dieser Dateiname kann entsprechend der SCP-Konvention aus maximal acht Zeichen fuer den eigentlichen Namen, gefolgt von einem Punkt "." und der bis zu drei Zeichen langen Namenerweiterung bestehen. Die Namenerweiterung wird auch als Dateityp bezeichnet.

REDABAS organisiert acht verschiedene Dateiarten und benutzt fuer die Typangabe Voreinstellwerte. Diese Voreinstellwerte sind fuer den Nutzer nicht zwingend, ihre Verwendung ist aber vorteilhaft, da dadurch bei Nennung des Dateinamens innerhalb von Befehlszeilen die explizite Angabe des Dateityps entfallen kann.
REDABAS arbeitet mit folgenden Dateitypen:

Datenbankdateien	.DBD
Indexdateien	.IDX
Variablendateien	.VAR
Befehlsdateien	.PRG
Reportdateien	.DEF
Maskendateien	.MSK
Textdateien	.TXT
Sicherungsdateien	.BAK

Diese Dateitypen werden im folgenden erlaeutert.

5.3.1. Datenbankdateien (.DBD)

Die Datenbankdatei ist das Kernstueck einer REDABAS-Datenbank. Sie enthaelt als erstes die Strukturbeschreibung der Datensaetze entsprechend der vom Nutzer vorgenommenen Definitionen im CREATE- oder MODIFY STRUCTURE-Befehl. Die Strukturbeschreibung kann maximal 32 Eintraege umfassen. Jeder Eintrag beschreibt ein Datenfeld und enthaelt die Angaben

- Feldname
- Datentyp
- Laenge des Datenfeldes
- Position des Datenfeldes im Satz.

Nach der Strukturbeschreibung sind in der Datenbankdatei die eigentlichen Datensaetze angeordnet. Ein Datensatz kann maximal 1000 Zeichen lang sein. Die Anzahl der Datensaetze darf 65 535 nicht ueberschreiten, eine Datei kann sich nur ueber einen Datentraeger erstrecken.

Datenbankdateien lassen sich aus bereits bestehenden Datenbankdateien durch Sortieren (SORT), Kopieren (COPY) oder Verknuepfen (JOIN) gewinnen.
Datenbankdateien sollten nicht mit Texteditoren oder aehnlichen Programmen bearbeitet werden!

5.3.2. Indexdateien (.IDX)

Indexdateien werden durch den INDEX-Befehl erzeugt und enthalten neben einer Eintragung zum indizierten Feld zu jedem Datensatz in der Datenbankdatei den Inhalt des Schluesselfeldes und einen Adressverweis zum zugehoerigen Datensatz.

Die Indizierung bewirkt die Sortierung der Datenbankdatei nach einem Schluessel, ohne dass eine Umordnung der Datenbankdatei erforderlich ist. Die Indexdatei beschleunigt den Zugriff auf einzelne Datensaetze vor allem bei groesseren Datenbankdateien.

5.3.3. Variablendateien (.VAR)

REDABAS legt eine Variablendatei an, wenn der Nutzer mit dem Befehl SAVE die in einer REDABAS-Sitzung erzeugten temporaeren Speichervariablen fuer die Nutzung in weiteren Programmlaeufen permanent speichern will. Die Variablendatei kann maximal 64 Speichervariablen -jede bis zu 254 Zeichen lang- aufnehmen. Eine Speichervariable kann Konstanten, Feldinhalte oder Ergebnisse von Berechnungen enthalten.

Durch den Befehl RESTORE wird die Variablendatei zur Wiederverwendung in den Arbeitsspeicher geladen.

5.3.4. Befehlsdateien (.PRG)

Haeufig benutzte Befehlsfolgen kann der Nutzer z.B. mit dem Befehl MODIFY COMMAND in einer Befehlsdatei speichern und mit dem DO-Befehl jederzeit zur Ausfuehrung bringen.

Solche Befehlsdateien koennen komplette Programme enthalten. Die Schachtelung von Befehlsdateien ist moeglich, d.h. in einer Befehlsdatei kann durch den DO-Befehl eine weitere Befehlsdatei aufgerufen werden. Die moegliche Schachtelungstiefe ist durch die maximale Anzahl von 16 gleichzeitig eroeffneten Dateien begrenzt, wobei jeder DO-Befehl das Eroeffnen der entsprechenden Befehlsdatei bewirkt. Ausserdem sind in der Zahl 16 auch die anderen momentan eroeffneten Dateien zu beruecksichtigen (z.B. Datenbankdateien, Indexdateien, Reportdateien).

Die Ausfuehrung einer Befehlsdatei wird beendet, wenn entweder das Ende der Datei erreicht ist oder der Befehl RETURN erreicht wird. Nach der Ausfuehrung steht der durch die Befehlsdatei belegte Platz im Arbeitsspeicher anderen Befehlsdateien zur Verfuegung.

Befehlsdateien koennen auch mit Editoren oder Textverarbeitungssystemen (z.B. "TEXT 30") erstellt und geaendert werden.

5.3.5. Reportdateien (.DEF)

Solch eine Datei erzeugt der REPORT-Befehl automatisch, wenn der Dialog zur Listengestaltung gefuehrt wird. Die Reportdatei enthaelt alle vom Nutzer waehrend des REPORT-Dialogs eingegebenen Angaben zur Struktur und zum Inhalt der gewuenschten Ergebnisliste.

Reportdateien koennen zwar mit Editoren oder Textverarbeitungssystemen geaendert werden, sinnvoller ist jedoch, die Reportdatei im REPORT-Dialog neu zu erstellen.

5.3.6. Maskendateien (.MSK)

Maskendateien enthalten ausschliesslich @-Befehle und *-Kommentare. Ein @-Befehl bewirkt die Einstellung auf eine bestimmte Cursor- oder Druckposition und ermoeglicht in Verbindung mit Parametern und nachfolgenden Befehlen eine programmierte Dateneingabe/ -ausgabe.

Durch den Befehl SET FORMAT TO <maskendatei> wird diese Datei im Arbeitsspeicher bereitgestellt und durch nachfolgende READ-, APPEND-, EDIT- oder INSERT-Befehle darauf zugegriffen. Bei Verwendung einer Maskendatei kann keine Druckausgabe mit SET FORMAT TO PRINT erfolgen.

Maskendateien koennen mit dem Befehl MODIFY COMMAND und der Angabe des Dateinamens einschliesslich des Datentyps erstellt werden. Ebenfalls lassen sich Textverarbeitungsprogramme dazu verwenden.

Wenn die @-Befehle direkt in die Befehlsdateien eingeordnet werden, ist die Verwendung von Maskendateien ueberfluessig.

5.3.7. Textdateien (.TXT)

Textdateien bestehen aus Saetzen, die sich aus Zeichen des ASCII-Zeichensatzes zusammensetzen. Jeder Satz ist entsprechend der SCP-Konvention mit RETURN und Zeilenvorschub abgeschlossen.

Die Verwendung von Textdateien bildet die Schnittstelle zwischen REDABAS und anderen Programmen.

Textdateien koennen als Eingabedateien im APPEND-Befehl im Zusammenhang mit dem Parameter "SDF" benutzt werden.

Der COPY-Befehl mit dem Parameter "SDF" erzeugt aus REDABAS-Datenbankdateien Ausgabedateien im ASCII-Format, die durch andere Programme weiterverarbeitet werden koennen.

Textdateien koennen auch das Ergebnis von SET ALTERNATE-Befehlen sein und stellen dann ein Protokoll von Bildschirmeingaben und -ausgaben dar.

5.3.8. Sicherungsdateien (.BAK)

Wird der Befehl MODIFY COMMAND benutzt, um eine Befehlsdatei, Maskendatei oder eine andere Standard-ASCII-Datei zu aendern, sichert REDABAS die vorhandene alte Version und vergibt dafuer den Dateityp .BAK.

Der Nutzer kann hier keinen eigenen Dateityp vergeben.

6. Hinweise zur Dialogfuehrung

6.1. Fehlerbehandlung

Mit dem Bereitschaftszeichen ":" auf dem Bildschirm fordert REDABAS den Nutzer zur Eingabe eines Befehls auf. Durch Betaetigung der <RETURN>-Taste (bzw. <ENTER>, <ET> oder <ET1>) wird die Befehlseingabe abgeschlossen. Bemerkt der Nutzer waehrend der Eingabe Tippfehler, kann er diese entweder durch die Korrekturtaste (Rueckwaertspfeil) verbessern oder mit der Tastenkombination <CONTROL-U> die gesamte Zeile loeschen und neu eingeben.

Nach erfolgter Eingabe prueft REDABAS die Befehlszeile auf korrekte Syntax und Semantik. Stellt das erste Wort kein gueltiges Befehlswort dar (es genuegt die Eingabe der ersten vier Zeichen des Befehlswortes), so erscheint die Bildschirmnachricht

UNBEKANNTES KOMMANDO!

Hat sich der Nutzer beim Installieren fuer den Fehlerkorrekturdialog entschieden, ist ein Ersetzen von ein oder mehreren Zeichen in der fehlerhaften Befehlszeile moeglich. Auf die Frage

"AENDERN VON:"

sind die zu ersetzenden Zeichen einzutippen und bei

"AENDERN AUF:"

die korrekten Zeichen einzugeben. Diese Moeglichkeit kann beliebig oft wiederholt werden, bis die Frage

"WEITERE KORREKTUREN? (J/N)"

verneint wird. Ist kein Fehlerkorrekturdialog einbezogen, muss die komplette Befehlszeile neu eingetippt werden. Der geuebte Bediener wird diese Variante bevorzugen.

Nach Pruefung des Befehlswortes erfolgt die Syntaxkontrolle der Befehlszeile. Erkennt REDABAS einen Fehler, erscheint die Meldung

SYNTAXFEHLER!

und ausserdem wird ueber das als ungueltig angesehene Wort in der nochmals angezeigten Befehlszeile ein Fragezeichen "?" gesetzt. Auch in diesem Fall kann eine Ersetzung der ungueltigen Zeichen erfolgen, wenn der Fehlerkorrekturdialog generiert ist.

Bei unvollstaendiger Befehlszeile, z.B. fehlendem Dateinamen, kann diese Information nach einer Aufforderung nachgereicht werden.

Weitere Fehler, die bei der Befehlsauswertung zutage treten, zeigt REDABAS mit spezifischen Fehlermeldungen an. Diese Fehlermeldungen sind am Ende der Programmtechnischen Beschreibung angefuehrt und erlaeutert. Auch in solchen Faellen kann bei eingeschaltetem Fehlerkorrekturdialog die Befehlszeile korrigiert werden.

Soll ein ausgefuehrter Befehl unmittelbar darauffolgend wiederholt werden, ist keine Neueingabe des Befehls erforderlich, es genuegt <CONTROL-R> einzutippen.

Eine "rollende" Bildschirmausgabe, die eine Seite uebersteigt, kann mit <CONTROL-S> angehalten und durch eine beliebige Taste fortgesetzt werden. Die <ESCAPE>-Taste bricht eine Bildschirmausgabe ab.

6.2. Seitenmodus und Cursorsteuerung

Die einfache Bildschirmausgabe erfolgt so, dass der Bildschirm von oben beginnend zeilenweise beschrieben wird. Der Cursor zeigt dabei die naechste freie Bildschirmposition an. Ist die letzte Bildschirmzeile beschrieben, so rueckt gewoehnlich das ganze Bild eine Zeile nach oben, wodurch die erste Zeile verschwindet und die letzte Bildschirmzeile fuer die naechste Anzeige frei wird. Man bezeichnet dies als "Rollen des Bildes".

Moderne Bildschirme besitzen die Moeglichkeiten der zeichenweisen Adressierung aller Bildschirmpositionen. Damit koennen die Rechnerprogramme ihre Ein- und Ausgaben auf dem Bildschirm gezielt plazieren. Auch REDABAS nutzt diese Moeglichkeiten und nennt diese Betriebsart, waehrend der der gesamte Bildschirm im Zugriff ist, "Seitenmodus". Befehle, die diesen Seitenmodus nutzen, sind APPEND, BROWSE, CREATE, EDIT, INSERT, MODIFY, @ und READ, vorausgesetzt der standardmaessig auf SET SCREEN ON gesetzte Parameter ist nicht ausgeschaltet (OFF).

Treten in diesen Befehlen ungewollte Bildschirmeffekte auf, kann eine falsche Bildschirmanpassung die Ursache dafuer sein (s. Abschnitt 2.2.).

Im Seitenmodus hat auch der Nutzer die Moeglichkeit, den Cursor an verschiedene Bildschirmpositionen zu bewegen, um den Inhalt von Datenfeldern sichtbar zu machen oder Dateneingaben zu bewerkstelligen.

Diese Steuerfunktionen sind ueber die Kombination der <CONTROL>-Taste und einer Buchstabentaste nutzbar.

Einige Tastenkombinationen sind allgemeiner Natur und unabhaengig von dem gerade aktiven Befehl. Andere haben in Abhaengigkeit vom verwendeten Befehl eine spezielle Bedeutung.

Bei einigen Tastenkombinationen gibt es Unterschiede zwischen der Version 1.0 und 2.0 von REDABAS. In der Version 2.0 sind die Cursortasten der Tastatur K 7637 unterstuetzt, damit wird die Cursorbewegung im Seitenmodus wesentlich erleichtert. Dadurch ergeben sich zwei notwendige Aenderungen fuer die Benutzung der <CONTROL>-Tastenkombination (Ein-Ausschalten der Loeschmarkierung; Verschieben des "Fensters" um ein Datenfeld zurueck).

In der folgenden Uebersicht ist die Bedeutung der Tastenkombinationen erlaeutert.

- Tastenkombinationen mit allgemeiner Bedeutung im Seitenmodus

CONTROL-G	Loeschen des durch den Cursor positionierten Zeichens
DELETE	Loeschen des Zeichens links vom Cursor
CONTROL-X CONTROL-F	Bewegen des Cursors zum naechsten Datenfeld
CONTROL-E CONTROL-A	Bewegen des Cursors zurueck zum vorhergehenden Feld
CONTROL-D	Bewegen des Cursors um ein Zeichen vorwaerts
CONTROL-S	Bewegen des Cursors um ein Zeichen zurueck
CONTROL-Y	Fuellen des rechten Teils des aktuellen Datenfeldes ab der Cursorposition mit Leerzeichen, bzw. Loeschen der Zeile rechts vom Cursor
CONTROL-V	Umschalten zwischen ueberschreibender und einfuegender Eingabe
CONTROL-Q	Abbrechen des Seitenmodus und Ueberleiten zum REDABAS-Befehlsmodus. Die letzten Aenderungen bleiben dabei unwirksam, d.h. sie werden nicht gespeichert.

- Tastenkombinationen fuer die Befehle APPEND, CREATE und INSERT

CONTROL-C CONTROL-R	Speichern des aktuellen Datensatzes auf die Diskette und Vorruecken zum naechsten Satz
RETURN	nur wenn der Cursor am Anfang des ersten Datenfeldes steht und am aktuellen Satz keine Aenderungen vorgenommen wurden: Beenden des Seitenmodus und Ueberleiten in den REDABAS-Befehlsmodus
CONTROL-U (Ver.1.0) CONTROL-L (Ver.2.0)	Setzen bzw. Loeschen der Loeschmarkierung "*" des aktuellen Datensatzes

- Tastenkombinationen fuer den Befehl EDIT

CONTROL-W	Schreiben des aktuellen Datensatzes auf die Diskette und Beenden der Funktion EDIT
CONTROL-C	Schreiben des (geaenderten) Datensatzes auf die Diskette und Aufrufen des naechsten Satzes zur Bearbeitung
CONTROL-R	Schreiben des (geaenderten) Satzes auf die Diskette und Aufrufen des vorhergehenden Satzes zur Bearbeitung
CONTROL-U (Ver.1.0) CONTROL-L (Ver.2.0)	Ein- / Ausschalten der Loeschmarkierung fuer einen Datensatz. Beim Einschalten erscheint am oberen Rand der Satzmaske das Wort GELOESCHT. Drueckt man die astenkombination nochmals, kann die Loeschung rueckgaengig gemacht werden.

- Tastenkombinationen fuer den Befehl BROWSE

CONTROL-W	Speichern des aktuellen Datensatzes auf die Diskette und Beenden des Seitenmodus
CONTROL-C	Bewegen des Cursors durch alle Felder mit gleichem Namen senkrecht nach unten
CONTROL-R	Bewegen des Cursors durch alle Felder mit gleichem Namen senkrecht nach oben zu bewegen
CONTROL-U (Ver.1.0) CONTROL-L (Ver.2.0)	Ein- / Ausschalten der Loeschmarkierung fuer einen Datensatz
CONTROL-B	Verschieben des "Fensters" um ein Datenfeld vorwaerts

CONTROL-Z (Ver.1.0) CONTROL-O (Ver.2.0) — Verschieben des "Fensters" um ein Datenfeld zurueck

- Tastenkombination fuer den Befehl MODIFY

CONTROL-N — Einfuegen eines neuen Datenfeldes bzw. einer Zeile an der Cursorposition; alle folgenden Datenfelder werden um eine Position nach hinten verschoben

CONTROL-T — Loeschen des Datenfeldes bzw. der Zeile an der Cursorposition; alle folgenden Datenfelder ruecken um eine Position vor

CONTROL-C — Anzeigen der folgenden Datenfelder bzw. Zeilen (naechste Bildschirmseite)

CONTROL-R — Anzeigen der vorangegangenen Datenfelder bzw. Zeilen (vorhergehende Bildschirmseite)

CONTROL-W — Speichern des aktuellen Datensatzes auf der Diskette und Beenden des Seitenmodus

- weitere allgemeine Steuerfunktionen

CONTROL-S — Stoppen und Fortsetzen einer Bildschirmausgabe

CONTROL-P — zusaetzliche Druckausgabe (hardcopy)

CONTROL-R — Wiederholung des zuletzt eingegebenen REDABAS-Befehls

CONTROL-U — Loeschen einer Befehlseingabe

ESCAPE — Abbruch einer Befehlsdatei bei schrittweiser Abarbeitung

In der Version 2.0 von REDABAS unter SCP (mit der Tastatur K7637) und in der Version 5.0 unter SCP1700 bestehen folgende funktionelle Uebereinstimmung:

Cursortaste (Version 2.0	Cursortaste (Version 5.0	Tastenkombination
<--	<-- , |<--	CONTROL-S
-->	-->	CONTROL-D
|<--	^	CONTROL-A bzw.
	|	CONTROL-E
-->|	|	CONTROL-F bzw.
	V	CONTROL-X
|		CONTROL-C bzw.
V		CONTROL-J
^		CONTROL-R bzw.
|		CONTROL-Z

7. Ausdruecke

Die in den REDABAS-Befehlszeilen verwendeten Ausdruecke koennen einfache Werte, aber auch Verknuepfungen von Werten sein. Die Auswertung eines Ausdrucks ergibt wieder einen einfachen Wert, der numerisch, logisch oder eine Zeichenreihe sein kann.

Werte koennen dargestellt werden durch

- Feldnamen
- temporaere Speichervariablen
- Konstanten.

Fuer die Verknuepfung und Berechnung von Werten bietet REDABAS

- Operationen
- Funktionen.

7.1. Variablen

Der Wert von Variablen ist veraenderlich und repraesentiert einen aktuellen Inhalt. Variablen werden stets mit Namen bezeichnet. REDABAS unterscheidet Datenbankvariablen und temporaere Speichervariablen. Datenbankvariablen sind alle durch den Feldnamen bezeichneten Felder eines aktuellen Datenbanksatzes. Temporaere Speichervariablen dienen als Zwischenspeicher und sind weder einer Datenbankdatei noch einem Datensatz zugeordnet.

Ihrem Typ nach koennen die Variablen numerisch, alphanumerisch oder logisch sein.

7.2. Konstanten

Konstanten sind im Sinne von REDABAS unveraenderliche Direktwerte, die nicht durch einen Namen bezeichnet werden, sondern selbsterklaerend sind.

Analog zu den Variablen gibt es numerische, alphanumerische und logische Konstanten.

Numerische Konstanten sind Zahlen, die Vorzeichen und Dezimalpunkt haben koennen (z.B. 1, -25, +21.7).

Alphanumerische Konstanten (=Zeichenreihen) muessen stets in Hochkommas ' ', Anfuehrungszeichen " " oder eckige Klammern [] eingeschlossen sein. Enthaelt eine Zeichenreihe eines dieser Zeichen, so ist die Konstante durch ein anderes der verbleibenden Sonderzeichen einzuschliessen.

Beispiele fuer alphanumerische Konstanten sind: [AUTO], 'Auto[s]',"a'b'c'd".

Logische Konstanten sind Wahrheitswerte und werden durch REDABAS wie folgt dargestellt:
T,t,J,j steht fuer "wahr" (englisch "true") bzw. "ja"
F,f,N,n steht fuer "falsch" (englisch "false") bzw. "nein"

7.3. Operationen

Aufgrund der von REDABAS unterstuetzten drei Datentypen ergeben sich folgende vier Arten von Operationen:

- arithmetische Operationen (Berechnungen)
- Vergleichsoperationen
- logische Operationen
- String-Operationen (Operation mit Zeichenreihen)

In einer Operation sind die Operanden durch Operatoren verknuepft. Ergebnisse von Operationen koennen selbst wieder Operanden sein. Zu diesem Zweck sind die Operationen ggf. in Klammern einzuschliessen. Durch Klammerung wird eine gewuenschte Ausfuehrungsreihenfolge von Operationen erreicht.

Bei der Formulierung von Operationen ist auf die Typengleichheit der Operanden und die Vertraeglichkeit von Operandentyp und Operatoren zu achten.

So kann z.B. nur eine Zeichenreihe mit einer anderen Zeichenreihe verknuepft werden, nicht aber eine Zeichenreihe mit einem numerischen Wert. In diesem Zusammenhang sei auf die Moeglichkeiten der Typumwandlung mittels Funktionen verwiesen (s. Abschnitt 7.4.).

Beispiele fuer Operationen:

```
: ? 3+8
 11

: ? 3+'a'
 SYNTAXFEHLER!

: ? 'a'+'b'
 ab
```

7.3.1. Arithmetische Operationen (Berechnungen)

Die Operanden von Berechnungen sind - ebenso wie die Ergebnisse - numerische Werte. Als arithmetische Operationen sind die vier Grundrechenarten zugelassen und durch folgende Operatoren anzugeben:

+	(Addition)
-	(Subtraktion)
*	(Multiplikation)
/	(Division)

Bei der Ausfuehrung der Operationen werden zunaechst Klammerausdruecke aufgeloest. Geschachtelte Klammern werden in der Reihenfolge von innen nach aussen berechnet.

Fuer die Rangfolge innerhalb der Operatoren gilt:
Multiplikation und Division vor Addition und Subtraktion.
Bei gleichrangigen Operatoren erfolgt die Auswertung von links nach rechts.

Beispiele:

```
: ? 3+4*5
 23

: ? (3+4)*5
 35

: ? ((9+4)*3+1)*2
 80

: ? (9+4)*3+1*2
 41
```

Operanden und Operatoren koennen durch Leerzeichen voneinander getrennt sein.

7.3.2. Vergleichsoperationen

Numerische Operanden und Zeichenreihen lassen sich vergleichen, wobei jeweils nur der gleiche Typ innerhalb eines Vergleichs auftreten darf.

Das Ergebnis einer Vergleichsoperation ist stets ein logischer Wert, .T. fuer "wahr" und .F. fuer "falsch".

Folgende Operatoren sind moeglich:

<	kleiner als
>	groesser als
=	gleich
<> oder #	ungleich
<=	kleiner oder gleich
>=	groesser oder gleich
¤	enthalten in (Substring-Suchoperator)

Der Substring-Suchoperator "¤" ist nur fuer Zeichenreihen verwendbar und testet, ob eine Zeichenreihe in einer anderen enthalten ist. Dieser Vergleich wird auch als gleitendes Suchen bezeichnet. Das Ergebnis ist wie bei allen Vergleichen ".T." oder ".F.".

Der Vergleich von Zeichenreihen erfolgt von links nach rechts, wobei in der Laenge des zweiten Operanden verglichen wird. Die Rangfolge der Zeichen ergibt sich aus der rechnerinternen Darstellung im ASCII-Code (in aufsteigender Reihenfolge: Leerzeichen, Sonderzeichen, Ziffern, weitere Sonderzeichen, Grossbuchstaben, weitere Sonderzeichen, Kleinbuchstaben, weitere Sonderzeichen).

Beispiele:

```
: ? 'MOND' < 'SONNE'
 .T.

: ? 3 > -5
 .T:

: ? 'ABC' > 'abc'
 .F.

: ? 'ab' = 'abc'
 .F.

: ? 'abc' = 'ab'
 .T.

: ? 'ERNST' ¤ 'BERNSTEIN'
 .T.
```

7.3.3. Logische Operationen

Solche Operationen sind nur auf Operanden, die logische Werte darstellen, anwendbar. Das Ergebnis ist ebenfalls ein logischer Wert.

Die moeglichen Operatoren in der Reihenfolge ihres Vorrangs sind:

```
    .NOT.   logische Verneinung
    .AND.   logisches UND
    .OR.    logisches ODER
```

Die Verwendung von Klammern ist moeglich, sie stehen in der Rangfolge der Abarbeitung ueber den anderen Operatoren.

Der .NOT.-Operator ist einstellig, d.h. er bezieht sich nur auf einen Operanden. Die logische Verneinung kehrt den Wahrheitswert des Operanden um.

Das logische UND ergibt den Wahrheitswert .T. nur dann, wenn beide Operanden "wahr" sind.

Das logische ODER bringt .T. als Ergebnis, wenn mindestens einer der Operanden "wahr" ist.

Beispiele:

```
: STORE F TO V1
 .F.

: STORE T TO V2
 .T.

: ? V1 .OR. V2
 .T.
```

```
: ? V1 .AND. V2
 .F.

: ? .NOT. V1 .AND. V2
 .T.
```

7.3.4. String-Operationen

String-Operationen werden auf Zeichenreihen angewendet und liefern auch als Wert Zeichenreihen. Das Ergebnis entsteht durch Aneinanderfuegen der Operanden. Die beiden moeglichen Operatoren unterscheiden sich wie folgt:

+ Zeichenreihen werden in der Reihenfolge ihrer Nennung in voller, definierter Laenge aneinandergefuegt.

- Zeichenreihen werden in der Reihenfolge ihrer Nennung aneinandergefuegt, wobei Leerzeichen am Ende der zu verbindenden Zeichenreihen an das Ende der Ergebniszeichenreihe verlagert werden.

Der "minus"-Operator "-" tilgt keine Leerzeichen, sondern sammelt diese und fuegt sie an das Ende der entstandenen Zeichenreihe an. Fuehrende oder eingebettete Leerzeichen innerhalb der Zeichenreihe bleiben unveraendert.

Die Beseitigung der Leerstellen vom Ende einer Zeichenreihe kann mit der Funktion "TRIM" erfolgen. (s. Abschnitt 7.4.)

Beispiele:

```
STORE 'Max       ' TO VNAME
 Max

STORE 'Meier     ' TO NAME
 Meier

? VNAME + NAME
 Max   Meier

? NAME-','-VNAME
 Meier,Max

? VNAME-NAME
 MaxMeier

? TRIM(VNAME)+' '+NAME
 Max Meier
```

7.3.5. Rangfolge von Operationen

Sind in einem Ausdruck mehrere Operationen enthalten, so erfolgt die Auswertung in einer definierten Reihenfolge.

Hierbei gelten Vorrangregeln, die entscheiden, welche Operation vor einer anderen ausgefuehrt wird. Treten

gleichrangige Operationen nebeneinander auf, so wird die Abarbeitung von links beginnend ausgefuehrt.

In der nachfolgenden Uebersicht sind die Operationen bzw. Operatoren ihrer Wertigkeit entsprechend aufgelistet, beginnend mit den ranghoechsten. Gleichrangige Operationen sind nebeneinander angeordnet:

- Klammern
- Funktionen
- Vorzeichen (+, -)
- Multiplikation, Division (*, /)
- Addition, Subtraktion (+, -)
- Vergleiche (<, >, =, <>, <=, >=, ¤)
- Stringoperationen (+, -)
- logische Verneinung (.NOT.)
- logisches UND (.AND.)
- logisches ODER (.OR.)

7.4. Funktionen

Bestimmte Verarbeitungsergebnisse lassen sich mit den einfachen arithmetischen, logischen oder Zeichenreihen-Operationen nur schwer oder gar nicht erreichen. REDABAS stellt deshalb eine Reihe von Algorithmen bereit, die vorprogrammierte Operationen enthalten und dem Nutzer als Funktionen zur Verfuegung stehen. Eine Funktion wird unter ihrem Namen aufgerufen, verarbeitet in der Regel das im Aufruf anzugebende Argument bzw. die Argumente und liefert im Ergebnis einen Funktionswert.

REDABAS unterscheidet nach dem Ergebnis (Funktionswert) drei Arten von Funktionen: numerische, alphanumerische und logische. Demzufolge sind die Ergebnisse von numerischen Funktionen Zahlenwerte, von alphanumerischen Funktionen Zeichenreihen, logische Funktionen ergeben die Wahrheitswerte "wahr" bzw. "true" oder "falsch" bzw. "false", angezeigt durch ".T." und ".F.". Fuer die Argumente gelten bestimmte Konventionen, die von der Art der Funktion abhaengig sind. Diese Regeln sind in der nachfolgenden Beschreibung der Einzelfunktionen erlaeutert. Verstoesse gegen diese Regeln ahndet REDABAS meist mit der Fehlermeldung "SYNTAXFEHLER!".

Im folgenden sind die Funktionen in einer Uebersicht zusammengestellt und anschliessend einzeln erklaert.

Funktionsuebersicht

Funktions-name		Bezeichnung
!	-	Grossbuchstabenfunktion
#	-	Satznummer-Funktion
¤	-	Substring-Funktion
*	-	Loeschabfrage-Funktion
@	-	Stringsuche-Funktion
CHR	-	Character-Funktion
DATE	-	Datum-Funktion
EOF	-	Dateiende-Funktion
FILE	-	Datei-Funktion
INT	-	Integer-Funktion
LEN	-	Laenge-Funktion
ORD	-	Ordinalitaet-Funktion
PEEK	-	Speicheranzeige-Funktion
STR	-	String-Funktion
TEST	-	Test-Funktion
TRIM	-	Trimm-Funktion
TYPE	-	Typ-Funktion
VAL	-	Wert-Funktion

Grossbuchstaben-Funktion

!(<zeichenreihen-ausdruck>)

Diese Funktion wandelt alle in der genannten Zeichenreihe vorkommenden Kleinbuchstaben (a...z) in Grossbuchstaben um. Alle anderen Zeichen in der Zeichenreihe bleiben unveraendert. Die Funktion wird unter dem Namen "!" aufgerufen. Das Argument kann Feldnamen, Variablen und in Hochkommas eingeschlossene Zeichenreihen und Kombinationen aus diesen enthalten.

Diese Funktion vereinfacht das Wiederauffinden von Daten, wenn die Erfassung in gemischter Gross-/Kleinschreibung erfolgte. Zu beachten ist, dass Umlaute, sofern diese auf der Tastatur verfuegbar sind, wie Sonderzeichen behandelt werden und nicht in Grossbuchstaben umgewandelt werden.

Beispiele:

```
: ? !('abc')
 ABC

: ? !('ab12'+'xy34')
 AB12XY34

: STORE 'kleinbuchstabe' TO VAR1
kleinbuchstabe

: ? !(VAR1)
KLEINBUCHSTABE
```

Satznummer-Funktion

#

Die Verwendung des Nummernzeichens als Funktionsname liefert eine ganze Zahl, die die aktuelle Satznummer der momentan aktiven Datenbankdatei verkoerpert.

Dadurch ist die aktuelle Datenbankposition jederzeit verfuegbar. Das Ergebnis kann auch den Wert "0" annehmen, wenn in einem vorausgegangenen FIND-Befehl kein Satz zur vorgegebenen Bedingung gefunden wurde.

Beispiele:

```
: ? #
     7

: SKIP+2
SATZ-NR. 00009

: ? #
     9
```

Substring-Funktion

¤(<zeichenreihen-ausdruck>, start>,<laenge>)

Mit dem internationalen Waehrungszeichen (auf manchen Tastaturen ist an dessen Stelle das Dollarzeichen zu finden), kann eine Funktion aufgerufen werden, die aus einer Zeichenreihe ein oder mehrere zusammenhaengende Zeichen heraussucht. Das Ergebnis ist eine eigene Zeichenreihe.

Zunaechst erarbeitet diese Funktion die Zeichenreihe gemaess des <zeichenreihen-ausdrucks> und entnimmt dieser, beginnend bei der durch <start> angegebenen Position, die durch <laenge> vorgegebene Anzahl von Zeichen.

<start> und <laenge> koennen selbst Ausdruecke sein, vorausgesetzt, diese Funktion wird nicht zur Bildung von Schluesselwerten fuer Indexdateien benutzt.

Bei Verwendung der Substring-Funktion fuer die Erzeugung von Schluesselwerten im INDEX-Befehl muessen fuer <start> und <laenge> Direktwerte (d.h Konstanten) angegeben werden.

Diese Funktion ist nicht mit dem Substring-Operator "¤" zu verwechseln, der eine andere Form des Aufrufs verlangt und einen Wahrheitswert ergibt (s. auch Abschnitt 7.3.2.)

Beispiele:

```
: ? ¤('abcdefgh',5,3)
 efg
```

```
: ? ¤('abc'+'def',3,2)
cd

: STORE 'abc123' TO VAR1
abc123
: STORE '456def' TO VAR2
456def
: STORE 4 TO BEGINN
4
: STORE 6 TO ANZAHL
6
: ? ¤(VAR1+VAR2,BEGINN,ANZAHL)
123456
```

Loeschabfrage-Funktion

*

Durch DELETE und andere Befehle zum Aendern von Datensaetzen kann in einen Datensatz eine Loeschmarkierung eingetragen werden. Mit dem Stern "*" wird eine logische Funktion aufgerufen, die den aktuellen Datensatz prueft, ob das Loeschkennzeichen eingetragen ist. Wenn ja, liefert die Funktion den Wahrheitswert ".T." (fuer true=wahr), andernfalls den logischen Wert ".F." (fuer false=falsch).

Beispiel:

```
: ? *
 .T.
```

Stringsuche-Funktion

@(<zeichenreihen-ausdruck1>,<zeichenreihen-ausdruck2>)

Diese Funktion prueft, ob die Zeichenreihe vom Ausdruck1 in der Zeichenreihe vom Ausdruck2 enthalten ist und liefert als Ergebnis eine ganze Zahl. Dieser numerische Wert bezeichnet die Position des Zeichens in der Zeichenreihe2, ab der die Uebereinstimmung mit der Zeichenreihe1 beginnt.

Der Vergleich erfolgt innerhalb der Zeichenreihen von links nach rechts. Es wird nur die erste Uebereinstimmung angezeigt. Wenn keine Uebereinstimmung vorliegt, hat das Ergebnis den Wert "0".

Beispiele:

```
: ? @('def','abcdefgh')
   4
7

: ? @('igel','zeisig'+'elch')
   5
```

Character-Funktion

CHR(<zahl>)

Mit der CHR-Funktion wird eine Zahl zwischen 0 und 255 in ein ASCII-Zeichen umgewandelt. Welches Zeichen aus welcher Zahl entsteht, bestimmt sich aus der Position im ASCII-Zeichensatz. Neben den druckbaren Zeichen kann man auf diese Weise auch Steuerzeichen fuer den Drucker oder Bildschirm erzeugen, die sich nicht immer ueber die Tastatur eingeben lassen. So ergibt sich beispielsweise aus CHR(13) ein Wagenruecklauf bzw. ein Ruecksetzen des Cursors auf den Zeilenanfang, durch CHR(10) erfolgt ein Zeilensprung. Mit CHR(12) kann man gewoehnlich den Bildschirm loeschen.

Somit ergibt sich die Moeglichkeit, in Nutzerprogrammen ueber Ausgabebefehle periphere Geraete zu steuern.

Beispiele:

```
: ? CHR(65)
A

: ? '1.Zeile'+CHR(13)+CHR(10)+'2.Zeile'
1.Zeile
2.Zeile
```

Datum-Funktion

DATE()

Mit dieser alphanumerischen Funktion erzeugt man eine Zeichenreihe, die in der Form TT.MM.JJ (Tag.Monat.Jahr) das bei Sitzungsbeginn oder ueber den SET DATE-Befehl eingegebene Datum enthaelt. Die Zeichenreihe ist 8 Zeichen lang. Beim Aufruf ist kein Argument anzugeben, die Klammern sollen nur die Verwechslung mit einer moeglichen Variablen DATE vermeiden.

Beispiel:

```
SET DATE TO 13.4.85
: ? DATE()
13.04.85
```

Dateiende-Funktion

EOF

Durch die logische Funktion EOF (End of File) kann geprueft werden, ob der aktuelle Datenbanksatz der letzte in dieser Datei war. War es der letzte, so liefert die Funktion den Wahrheitswert ".T." (=wahr), andernfalls den Wert ".F." (=falsch). EOF wird erst dann auf ".T." gesetzt, wenn ein Zugriff ueber den letzten Satz hinaus versucht wird.

Diese Funktion kann in Programmschleifen verwendet werden, wenn eine Datei satzweise, zyklisch verarbeitet wird.

Beispiele:

```
: GO TOP
: ? EOF
 .F.

: GO BOTTOM
: ? EOF
 .F.
: SKIP
SATZ-NR. 00020
: ? EOF
 .T.
```

Datei-Funktion

FILE(<dateiname>)

Mit dieser logischen Funktion kann man pruefen, ob die in <dateiname> angegebene Datei auf der Diskette vorhanden ist oder nicht. <dateiname> kann durch eine Zeichenreihe, eine Variable oder auch durch einen Ausdruck dargestellt werden.

Sind das Laufwerk und der Dateityp nicht angegeben, so gelten das aktuelle Laufwerk und der Dateityp .DBD als Standardwerte.

Ist die gesuchte Datei vorhanden, liefert die Funktion den Wert ".T." (=wahr), andernfalls den Wert ".F." (=falsch).

Durch diese Funktion kann man innerhalb von Programmen einen erforderlichen Diskettenwechsel erkennen.

Beispiele:

```
: ? FILE('KUNDEN')
 .T.

: STORE 'B:' TO LW
B:
: ? FILE(LW + 'KUNDEN.IDX')
 .F.
```

Integer-Funktion

INT(<numerischer ausdruck>)

Diese numerische Funktion liefert den ganzzahligen Teil des <numerischen ausdrucks>. Dabei wird zunaechst der Ausdruck errechnet und dann der Dezimalpunkt und die nachfolgenden Stellen abgeschnitten. Es erfolgt keine Rundung, im Bedarfsfall kann dies je nach Vorzeichen durch eine Addition bzw. Subtraktion (z.B. mit 0.5) erreicht werden.

Beispiele:

```
: ? INT(12345.678)
  12345

: STORE 65.78 TO RECH
 65.78

: ? INT(RECH + 0.5)
        66
```

Laenge-Funktion

LEN(<zeichenreihen-ausdruck>)

Mit dieser numerischen Funktion kann man die Zeichenanzahl von Zeichenreihen ermitteln. Fuer Feldnamen und Variablen liefert die Funktion die definierte Laenge, die aktuelle Anzahl der Zeichen kann hier geringer sein.

Diese Funktion eignet sich zur Ermittlung des Platzbedarfs von Informationen auf einer selbst programmierten Liste.

Beispiele:

```
: ? LEN('abcdefg')
   7

: STORE 'Fleischer' TO V1
Fleischer
: STORE 'Max' TO V2
Max
: ? LEN(V1 + V2)
  12
```

Ordinalitaet-Funktion

ORD (<zeichenreihen-ausdruck>)

Die ORD-Funktion wandelt das 1. Zeichen der angegebenen Zeichenreihe in eine ganze Zahl um. Der Ergebniswert ergibt sich aus der Stellung (Ordinalitaet) des Zeichens im ASCII-Alphabet und entspricht damit der rechnerinternen Darstellung.

Beispiele:

```
: ? ORD('A')
65

: ? ORD('a')
97

: ? ORD('abc')
97
```

Speicheranzeige-Funktion

PEEK (<adresse>)

PEEK ist eine numerische Funktion und liefert den Wert , der unter der angegebenen <adresse> im Arbeitsspeicher steht. Die <adresse> und das Ergebnis sind Dezimalzahlen. PEEK zeigt keinen korrekten Inhalt vom Speicherbereich 1000H - 7A00h an.

Beispiel:

```
: ? PEEK(256)
 195

: STORE 42752 TO V1
: ? PEEK V1
 229
```

String-Funktion

STR(<numerischer ausdruck>,<laenge>,[<dezimalstellen>])

STR ist eine alphanumerische Funktion, die eine Zahl in eine Zeichenreihe umwandelt. Bei Kommazahlen sind bei <laenge> der Dezimalpunkt und die Anzahl der gewuenschten Dezimalstellen mit einzubeziehen. <dezimalstellen> enthaelt die Anzahl der gewuenschten Stellen nach dem Dezimalpunkt.

Fuer <laenge> und <dezimalstellen> sind Konstanten, Variablen und Ausdruecke zugelassen. Ist der numerische Wert kuerzer als <laenge>, so wird der Rest mit Leerzeichen aufgefuellt. Besitzt der ganzzahlige Teil des numerischen Wertes mehr Stellen als in <laenge> angegeben sind, wird das Ergebnis mit *** angegeben.

Die STR-Funktion wird benutzt, um Zeichenreihen-Operationen mit numerischen Feldern ausfuehren zu koennen.

Bei Verwendung dieser Funktion im INDEX-Befehl zur Bereitstellung des Schluessels sind <laenge> und <dezimalstellen> als Konstanten anzugeben.

Beispiele:

```
: ? STR(378.456,5,1)
  378.4

: STORE 325.896 TO ZAHL
 325.896

: STORE STR(ZAHL,6,2) TO ZEICHEN
325.89

: DISPLAY MEMORY
ZAHL        (N)       325.896
ZEICHEN     (C)       325.89
** SUMME **     02 TEMP. VARIABLE, 0014 BYTES LANG
```

Test-Funktion

TEST(<ausdruck>)

Diese Funktion kann dazu verwendet werden, vom Nutzer erklaerte Ausdruecke vor deren Verwendung auf syntaktische Richtigkeit zu pruefen. Dadurch vermeidet man moegliche Fehlerausschriften.

Wenn der Ausdruck korrekt ist, liefert die Funktion einen von Null verschiedenen Wert, der aber nicht dem Wert des Ausdruckes entspricht. Bei fehlerhaftem Ausdruck ist das Ergebnis "0" (=NULL).

Beispiel:

```
: ? TEST('abc' + 123)
   0

: ? TEST('abc' + 'cde')
   6

: ? TEST(123 + 345)
  -6
```

Trimm-Funktion

TRIM(<zeichenreihen-ausdruck>)

Die Funktion TRIM entfernt die Leerstellen, die sich am Ende einer Zeichenreihe befinden koennen. Das Ergebnis dieser alphanumerischen Funktion ist die um die Leerstellen gekuerzte Zeichenreihe.

Der <zeichenreihen-ausdruck> kann Konstanten, Variablen oder Feldnamen enthalten.

Fuer die Bereitstellung von Schluesselwerten im INDEX-Befehl darf diese Funktion nicht benutzt werden, da REDABAS mit konstanten Schluessellaengen rechnet.
Beispiele:

```
: STORE 'AB CD   ' TO VAR1
AB CD

: ? LEN(VAR1)
   8

: STORE TRIM(VAR1) TO VAR2
AB CD

: ? LEN(VAR2)
  5
```

Typ-Funktion

TYPE(<ausdruck>)

Diese alphanumerische Funktion bestimmt den Datentyp des angegebenen Ausdruckes und liefert als Wert ein Zeichen. Folgende Zeichen sind moeglich: "C" fuer Zeichenreihen-Ausdruecke, "N" fuer numerische Ausdruecke, "L" fuer logische Ausdruecke und "U" fuer undefinierte Variablen oder Feldnamen. Das Ergebnis der Funktion wird in Grossbuchstaben bereitgestellt.

Beispiele:

```
: STORE 'abc' TO VAR1
abc
: ? TYPE(VAR1)
C
: STORE 123 TO VAR2
  123
: ? TYPE(VAR2)
N
```

Wert-Funktion

VAL(<zeichenreihen-ausdruck>)

VAL ist eine numerische Funktion und wandelt den <zeichenreihen-ausdruck> in einen numerischen Wert um. Die Zeichenreihe darf nur aus Ziffern und wahlweise einem Vorzeichen und einem Dezimalpunkt bestehen. Bei der Umwandlung werden der Dezimalpunkt und nachfolgende Stellen abgeschnitten. Die VAL-Funktion liefert immer einen ganzzahligen Wert. Enthaelt die Zeichenreihe nur am Anfang Ziffern, so ergibt sich das Ergebnis nur aus diesen fuehrenden Ziffern. Sind keine fuehrenden Ziffern vorhanden, ist das Ergebnis "0". Fuehrende Nullen werden durch die VAL-Funktion geloescht.

VAL kann auch zusammen mit der Substring-Funktion verwendet werden, wenn nur Teile einer Zeichenreihe als Zahl dargestellt werden sollen.

Eine Moeglichkeit, den Dezimalpunkt und nachfolgende Stellen bei der Umwandlung von Zeichenreihen in Zahlen zu erhalten, bietet die Makroersetzung (s. Abschnitt 7.5.).

Beispiele:

```
: ? VAL('345.678')
     345

: ? DATE()
13.04.85

: ? VAL(DATE())
       13
```

```
: ? VAL(¤('abc123def',4,3))
 123
: STORE '456.789' TO VAR1
  456.789

: ? 500 + &VAR1
  956.789

: ? VAL('00012')
      12
```

7.5. Makros

REDABAS bietet die Moeglichkeiten, die wiederholte Eingabe von Zeichenreihen zu vermeiden bzw. bedeutend zu verkuerzen, indem stellvertretend fuer solch eine Zeichenreihe der Name einer temporaeren Speichervariablen steht, der ein definiertes Kennzeichen vorangestellt wird. Dieses Kennzeichen fuer Makroersetzungen ist das "&" (= Ampersand).

Trifft REDABAS in Befehlen oder bei der Auswertung von Ausdruecken auf das Zeichen "&", so werden die darauffolgenden Zeichen - wenn moeglich - als Name einer temporaeren Variablen gedeutet, und REDABAS ersetzt an derselben Stelle den Variablennamen einschliesslich des "&" durch die unter dem Namen gespeicherte Zeichenreihe.

Auf diese Weise koennen ganze Befehlszeilen durch die Angabe "&" und eines Variablennamen aufgerufen werden. Die Befehle ELSE, ENDIF und ENDDO koennen nicht durch Makros ersetzt werden!

Ebenso lassen sich ueber solche Makros Parameterwerte zwischen Befehlsdateien austauschen.

Sollen hinter einer Zeichenreihe, die durch eine Makroersetzung gebildet wird, noch weitere Zeichen angefuegt werden, so muss der Variablenname mit einem Punkt "." von den nachfolgenden Zeichen abgegrenzt werden. REDABAS versucht, alle Zeichen zwischen dem Zeichen "&" und dem naechsten Sonderzeichen (ausser :) oder Leerzeichen als Variablennamen zu deuten. Der Doppelpunkt wird nicht als Begrenzung akzeptiert, weil er Bestandteil eines Variablennamen sein kann. Wird ein anderes Sonderzeichen anstelle des Punktes "." zur Abgrenzung verwendet, so bleibt dieses Zeichen Bestandteil der Zeichenreihe. Folgt dem Zeichen "&" kein Name einer temporaeren Variablen, findet keine Ersetzung statt und das Zeichen "&" verbleibt im Ausdruck.

Anstelle des Zeichens "&" kann der Nutzer im Rahmen der Installierung von REDABAS auch ein anderes Sonderzeichen fuer die Makroersetzung benennen.

Beispiele:

```
: STORE 'USE BESTAND' TO B1
 USE BESTAND

: STORE 'DELETE RECORD' TO B2
 DELETE RECORD


: &B1                    - Eroeffnen der Datei "BESTAND"
: &B2 4                  - Loeschen des Datensatzes Nr. 4
00001 SATZ(SAETZE) GELOESCHT
: STORE 'BAHN' TO V1
 BAHN

: ? 'AUTO&V1'
 AUTOBAHN

: ? '&V1.HOF'
 BAHNHOF

: ? '&V1-HOF'
: BAHN-HOF

: ? '&V1HOF'
: &V1HOF
```

8. Befehle

8.1. Allgemeines

Bevor im Abschnitt 8.3. eine Zusammenstellung aller REDABAS-Befehle - alphabetisch geordnet - und deren ausfuehrliche Beschreibung geboten wird, sollte man sich mit den Grundregeln der REDABAS-Kommandosprache und den im Abschnitt 8.2. dargestellten Symboldefinitionen vertraut machen.

Im Abschnitt 2.3. wurden bereits einige Grundregeln dargelegt, die fuer das Verstaendnis und Nachvollziehen der Funktionen der Datenbankverwaltung und der Programmierung mit REDABAS unbedingt notwendig sind. Um in diesem Kapitel eine lueckenlose Darstellung aller Befehle, ihrer Beschreibung und Wirkungsweise zu bieten, werden gegebenfalls Wiederholungen in Kauf genommen.

- Ein Befehl kann (im Sinne von REDABAS) aus dem Befehlswort (z.B. CREATE, LIST, WAIT) und Zusatzklauseln, die die durch das Befehlswort eingeleitete Aktion spezifizieren und mit den Woertern FOR, NEXT oder WITH beginnen, bestehen. Befehlswoerter und Einleitungswoerter von Zusatzklauseln sind die "Steuerwoerter" von REDABAS.

- In einer Befehlszeile muss das erste vom Leerzeichen unterschiedliche Zeichen zum Befehlswort gehoeren. Zusatzklauseln sind in ihrer Reihenfolge wahlfrei.

- Die maximale Laenge einer Befehlszeile ist 254 Zeichen. Dabei muessen Makro-Ersetzungen bereits enthalten sein. Befehle, die in ihrer Laenge eine Bildschirmzeile ueberschreiten, koennen bei Einhaltung der Maximallaenge von 254 Zeichen auf mehrere Bildschirmzeilen aufgeteilt werden. Dabei muss jeder Teil des Befehls am Ende der Zeile mit Semikolon abgeschlossen werden.

- Teile eines Befehls koennen untereinander durch beliebig viele Leerzeichen getrennt sein. Diese Leerzeichen sind bei der Maximallaenge mit zu beruecksichtigen.

- Steuerwoerter, Feldnamen, Namen temporaerer Speicher-variablen, Dateinamen koennen gross oder klein eingegeben werden.

- Steuerwoerter werden durch ihre ersten vier Buchstaben eindeutig identifiziert, d.h. eine entsprechende Abkuerzung ist zulaessig. Werden mehr als vier Buchstaben angegeben, muss diese Schreibweise korrekt der Buchstabenfolge des vollstaendigen Steuerwortes entsprechen (MODIFY COMMAND waere durch MODI COMM oder MODIF COMMA eindeutig).

- Sogenannte Zusaetze oder Optionen sind in eckige Klammern ([...]) gesetzt und koennen wahlfrei verwendet werden.

- Symbole sind in spitze Klammern (<...>) eingeschlossen und muessen entsprechend ihrer Bedeutung (s. Abschnitt 8.2.) durch echte Daten ersetzt werden.

- Um Verwechslungen zu vermeiden, ist es zweckmaessig, REDABAS-Steuerwoerter nicht als Namen fuer Dateien, Felder oder Speichervariablen zu verwenden.

8.2. Symboldefinitionen

<adresse> - absolute Adresse im Arbeitsspeicher (Angabe dezimal)

<ausdruck> - eine - nach den Regeln von REDABAS - sinnvolle Verknuepfung von Konstanten, Feldnamen, temporaeren Speichervariablen, Funktionen und Operationen

<ausdrucksfolge> - durch Kommas getrennte Folge von <ausdruecken>

<bedingung> - Ausdruck, der als Auswahlkriterium fuer Dateiverwaltung, Dateizugriff oder fuer Programmfortsetzung dient

Neben der Einleitung einer Bedingung mit FOR ist im allgemeinen auch die Angabe von WHILE moeglich.
WHILE <bedingung> betrifft alle Saetze, die nach dem aktuellen Datensatz kommen, so lange, bis die <bedingung> nicht mehr erfuellt ist.
FOR und WHILE sollten nicht gemeinsam benutzt werden, mehrere Bedingungen koennen aber logisch verknuepft werden.

<befehl> - ein oder mehrere gueltige REDABAS-Befehle

<befehlsdatei> - Name einer Datei vom Typ .PRG zur Speicherung von Befehlsfolgen

<begrenzung> - beliebiges Sonderzeichen (ausser ";"), in das Felder vom Typ <zeichenreihe> eingeschlossen werden

<bereich> - Bereich der Datei, fuer den der verwendete Befehl gilt. Zulaessig sind folgende Angaben

ALL - es werden alle Saetze einer Datei, unabhaengig von der aktuellen Datensatznummer beruecksichtigt

NEXT <zahl> - betrifft die naechsten, durch <zahl> spezifizierten Datensaetze und den aktuellen Satz

RECORD <zahl> - betrifft nur den Datensatz mit der durch <zahl> spezifizierten Nummer

<bytefolge> - Folge von Dezimalzahlen von 0 bis 255, durch Kommas getrennt

<dateiname> - beliebiger Dateiname entsprechend der SCP-Konvention, der sich aus max. 8 Zeichen fuer den Namen, einen Punkt und max.

3 Zeichen fuer den Dateityp zusammensetzt. REDABAS vergibt fuer verschiedene Dateiarten automatisch den Dateityp (s. Abschn. 5.1.).

<datum> - Datumangabe in der Form TT.MM.JJ (Tag. Monat.Jahr) oder MM.TT.JJ (entsprechend der Installierung)

<dezimalstellen> - Anzahl Ziffern nach dem Dezimalpunkt einer Dezimalzahl

<feldname> - Name eines beliebigen Datenfeldes (max. 10 Zeichen lang)

<feldnamen> - mehrere, durch Kommas getrennte Namen von Datenfeldern

<indexdatei> - Name einer Datei vom Typ .IDX zur Speicherung eines Zugriffsindex (wird durch den INDEX-Befehl erzeugt)

<koordinaten> - Bildschirm- oder Druckposition in der Form x,y (x=Zeilennummer,y=Spaltennummer)

<laenge> - Laenge oder Teillaenge eines Feldes (ganze Zahl)

<laufwerk> - Bezeichnung des Diskettenlaufwerkes (z.B. A:, B:)

<maske> - Zeichenfolge, die als Schablone fuer die Dateneingabe/-ausgabe dient

<maskendatei> - Name einer Datei vom Typ .MSK, die Masken zur Dateieingabe/-ausgabe enthaelt

<muster> - <dateiname> oder Name einer temporaeren <speichervariablen> mit Ersetzungszeichen ("?" fuer einzelne Zeichen, "*" fuer den Rest des Namens)

<numerischer ausdruck> - Ausdruck, dessen Auswertung einen Zahlenwert ergibt

<parameter> - Beiwoerter zur naeheren Spezifizierung von Befehlen

<reportdatei> - Name einer Datei vom Typ .DEF zur Speicherung einer Listenstruktur, die durch den REPORT-Befehl erzeugt wird.

<schluessel> - Feldname, der als Ordnungskriterium einer Datenbankdatei dient

<SCP-Befehlsfolge> - durch Kommas getrennte Folge von SCP-Kommandos bzw. -Programmnamen, die in Hochkommas einzuschliessen sind

<speichervariable(n)> - Name von einer bzw. mehreren beliebigen temporaeren Variablen zur Zwischenspeicherung von Werten fuer die spaetere Wiederverwendung.
Mehrere Namen sind durch Kommas zu trennen.

<start> - Position im Datenfeld (mit 1 beginnend)

<taste> - einer der symnolischen Namen f1, f2...f10 fuer die entsprechenden Steuertasten FP1, PF2...PF10 der Tastatur des A7100

<text - beliebiger Text

<variable> - Name eines beliebigen Datenfeldes oder einer <speichervariablen>

<zahl> - ganze, positive Zahl

<zeichen> - ASCII-Zeichen, das in Hochkommas eingeschlossen wird

<zeichenreihe> - beliebige Folge von <zeichen>, die in '', "" oder [] eingeschlossen sein kann

<zeichenreihen-ausdruck> - Ausdruck, dessen Auswertung eine <zeichenreihe> ergibt

8.3. Beschreibung der Befehle

?

a) ? [<ausdrucksfolge>]
b) ?? <ausdrucksfolge>

Dieser Befehl entspricht dem Befehl DISPLAY <ausdrucksfolge> OFF und dient dem Anzeigen eines Ausdrucks oder einer Ausdrucksfolge. DISPLAY setzt im Unterschied zu dem ?-Befehl ein USE voraus.

Eine <ausdrucksfolge> kann Konstanten, Funktionen, Feldnamen und temporaere Speichervariablen enthalten.

Form a) dieses Befehls bewirkt vor der Ausgabe der <ausdrucksfolge> einen Zeilenvorschub auf dem Ausgabemedium. Ist keine <ausdrucksfolge> angega) kein Zeilenvorschub vor der Ausgabe der <ausdrucksfolge> ausgegeben.

Beispiel:

```
: ? 5*7
  35
```

@

@ <koordinaten> [SAY <ausdruck> [USING <maske>]]
[GET <variable> [PICTURE <maske>]]__

Auf manchen Tastaturen ist statt des "@" (Klammeraffe das Pragraph-Zeichen zu finden und zu benutzen.

Dieser Befehl ist ein wichtiges Hilfsmittel fuer die programmierte Dateneingabe und -ausgabe. In Verbindung mit den Befehlen ERASE, EJECT, CLEAR GETS, READ, SET FORMAT TO und weiteren Programmparametern hat der Nutzer die Moeglichkeit, sowohl Ausgaben auf dem Drucker oder Bildschirm nach eigenen Wuenschen festzulegen als auch Eingaben in Datenfelder oder Speichervariablen zu bewerkstelligen. Die Wirkung des @-Befehls und der anderen oben angefuehrten Befehle wird wesentlich durch den vorausgegangenen SET FORMAT-Befehl beeinflusst.

Mit <koordinaten> ist im @-Befehl immer ein Wertpaar anzugeben, das eine Druckposition oder die Cursorposition auf dem Bildschirm beschreibt. Die Angabe von <koordinaten> erfolgt in der Form "Zeilennummer, Spaltennummer", wobei die Werte mit "0" beginnen. Die linke obere Ecke auf dem Bildschirm bzw. auf einer Druckliste hat somit die Koordinaten "0,0", die Angabe "12,15" bezeichnet die 13.Zeile und 16.Spalte. Die 1.Zeile auf dem Bildschirm sollte nicht benutzt werden, da sie fuer REDABAS-Nachrichten reserviert ist. Der

Wertevorrat fuer <koordinaten> richtet sich nach der Bildschirmgroesse bzw. dem gewuenschten Druckformat. Fuer die Drucksteuerung sind Spaltennummern zwischen 0 und 254 moeglich, die Zeilennummer darf 127 nicht uebersteigen.

<koordinaten> koennen durch Konstanten, numerische Variablen oder numerische Ausdruecke angegeben werden.

Ob sich die Positionierung und nachfolgende Ein- /Ausgabe auf den Bildschirm oder auf den Drucker bezieht, bestimmt der Nutzer zuvor durch einen SET FORMAT -Befehl.

Ein SET FORMAT TO SCREEN-Befehl, der der Standardeinstellung entspricht, bestimmt den Bildschirm als Ein- /Ausgabegeraet fuer alle nachfolgenden @-Befehle.

Ein ERASE-Befehl loescht den gesamten Bildschirm, entaktiviert alle GET-Klauseln und setzt den Cursor in die linke obere Ecke des Bildschirms. Enthaelt der @-Befehl keine SAY- oder GET-Klauseln, wird nur der Rest der Bildschirmzeile ab der vorgegebenen Cursorposition geloescht, d.h. mit Leerzeichen gefuellt.

Bei Bildschirmausgabe koennen die <koordinaten> auch relativ angegeben werden, z.B. @ ¤, ¤+2. Mit dem Zeichen "¤" (auf manchen Tastaturen steht dafuer auch das Dollar-Zeichen) wird die Bildschirmposition angesprochen, die durch den vorangegangenen @-Befehl und der mit ihm verbundenen Ausgabe erreicht wurde.

Ein SET FORMAT TO PRINT-Befehl bewirkt die Druckausgabe der nachfolgenden @-Befehle. Im Gegensatz zur Bildschirmausgabe muessen hier die @-Befehle in aufsteigender, nach Zeilen- und Spaltennummer geordneter Folge genannt werden, ansonsten kann ungewollt ein Seitenvorschub erfolgen, wenn die Folgekoordinaten niedriger sind als die vorausgegangenen.

Mit dem EJECT-Befehl wird ein Seitenvorschub des Druckers erreicht und eine Rueckstellung auf die Koordinaten 0,0.

Die wahlweise SAY-Klausel ermoeglicht die Ausgabe eines <ausdrucks> an der durch <koordinaten> vorgegebenen Position auf dem durch SET FORMAT TO ... bestimmten Ausgabemedium. Der <ausdruck> kann unabhaengig sein oder aber die Erlaeuterung zu einer folgenden GET-Klausel darstellen.

Die GET-Klausel ist nur fuer Bildschirmausgabe wirksam. Sie gibt den Inhalt des mit <variable> bezeichneten Datenfeldes oder der Speichervariablen auf den Bildschirm aus und ermoeglicht mit einem anschliessenden READ-Befehl, diesen Wert ueber die Tastatur zu aendern und den neuen Inhalt der <variablen> zuzuweisen.

Es besteht die Moeglichkeit, nach jedem @...GET...-Befehl einen READ-Befehl auszufuehren, aber man kann auch maximal 64 GET-Klauseln gleichzeitig aktivieren, bevor durch READ die Aenderung der angezeigten Variablen erfolgt. REDABAS merkt sich die Bildschirmposition der einzelnen Felder, positioniert bei Ausfuehrung des READ-Befehls auf das zuerst genannte Feld und bewegt den Cursor bei <RETURN> auf das naechste.

Es wirken auch hier die <CONTROL-/Buchstaben>-Tasten zur Cursorsteuerung.

Ist der Programmparameter SET COLON ON gesetzt, werden die Feldgrenzen von GET-<Variablen> durch Doppelpunkte auf dem Bildschirm markiert.

Durch die Befehle ERASE oder CLEAR GETS werden alle aktiven GET-Klauseln entaktiviert.

Fuer die Ausgabe von Ausdruecken durch SAY, fuer die Anzeige von Variablen mit GET und fuer die Eingabe von Variablen mit READ koennen <masken> benutzt werden. Fuer SAY dient dazu die USING-Klausel. Die angegebene <maske> erzeugt die gewuenschte Ausgabeaufbereitung und verhindert u.U. ungewollte Seiteneffekte. Der moegliche Maskenaufbau ist untenstehend erlaeutert.

Bei GET kann in der PICTURE-Klausel eine <maske> angegeben werden. Diese dient sowohl der Anzeigeaufbereitung der GET-Variablen als auch der Plausibilitaetspruefung fuer eine nachfolgende Eingabe mit dem READ-Befehl. So kann z.B. durch die <maske> angezeigt werden, ob numerische Zeichen oder Alphazeichen einzugeben sind.

Es ist darauf zu achten, dass die <masken> fuer numerische Variable ausreichend dimensioniert sind. Ist das nicht der Fall, werden bei der Anzeige entweder Dezimalstellen abgeschnitten oder durch "*" ein numerischer Ueberlauf angezeigt.

Maskenaufbau:

Die <maske> in der USING-Klausel beschreibt die Darstellung einer Ausgabe. Die <maske> in der PICTURE-Klausel beschraenkt darueber hinaus noch die zulaessigen Eingabezeichen. Eine <maske> besteht aus einer Folge von Zeichen, die in Hochkommas einzuschliessen sind. Die unterschiedliche Bedeutung der Zeichen zeigt nachstehende Tabelle

Die verschiedenen Moeglichkeiten zeigt nachstehende Tabelle:

Masken-zeichen	Bedeutung fuer USING	Bedeutung fuer PICTURE
# oder 9	bewirkt die Ausgabe einer Ziffer des <ausdrucks> anstelle des Masken-zeichens oder die Ausga-be eines Leerzeichens, wenn der Ausdruck an die-ser Stelle keine Ziffer enthaelt	erlaubt nur die Eingabe einer Ziffer (0...9), eines Vorzeichens (+,-), oder eines Leerzeichens
X	bewirkt die Ausgabe eines beliebigen ASCII-Zeichens	erlaubt die Eingabe eines beliebigen ASCII-Zeichens

A	-	erlaubt nur die Eingabe von Buchstaben
¤ oder *	bewirkt die Ausgabe einer Ziffer oder die Ausgabe "¤" oder "*" anstelle einer fuehrenden Null	-
!	-	wandelt sowohl bei der Ausgabe als auch bei der Eingabe Kleinbuchstaben in Grossbuchstaben um

Hinweis: Die Maskenzeichen "X" und "A" muessen als Grossbuchstaben geschrieben werden. Andere als die zulaessigen Maskenzeichen fuehren bei der Ausgabe nur zur Darstellung der Maskenzeichen und zeigen nicht den Inhalt des <ausdrucks> bzw. der <variablen> an.

Andere als die zulaessigen Maskenzeichen werden bei einer GET <variable> PICTURE <aske> - Klausel als Feldinhalt in die <variablen> uebernommen und lassen sich in einem nachfolgenden READ-Befehl nicht veraendern. Bei Eingabe einer Dezimalzahl ist der Dezimalpunkt bereits in der Maske zu definieren.

Beispiel:

```
@ 14,1 SAY 'Namen eingeben:'
@ 2,5 SAY 'Preis:          M'
@ 2,12 SAY preis USING '***#.##'
```

(und s. Abschnitt 4.1.)

ACCEPT ['<text>'] TO <speichervariable>

Dieser Befehl ermoeglicht die Eingabe beliebiger alphanumerischer Ausdruecke von der Tastatur in eine <speichervariable>. Die Eingabe braucht - im Gegensatz zur Eingabe bei INPUT - nicht in Anfuehrungszeichen eingeschlossen werden, da ACCEPT jede Eingabe als alphanumerischen Wert auffasst.

Ist die <speichervariable> noch nicht vorhanden, wird sie angelegt und die Eingabe des Benutzers dort gespeichert. Erfolgt keine Eingabe, d.h. wird nur die RETURN-Taste gedrueckt, enthaelt die <speichervariable> ein Leerzeichen.

Vor der Eingabe kann man eine Erlaeuterung ueber die Art der zu erwartenden Daten geben. Dieser <text> wird in Anfuehrungszeichen, Hochkommas oder eckige Klammern eingeschlossen, wobei beiderseitig das gleiche Zeichen zu verwenden ist. Er erscheint, von einem Doppelpunkt gefolgt, auf dem Bildschirm und stellt fuer den Nutzer die Aufforderung zur Eingabe dar.

Beispiel:

```
ACCEPT "BITTE GEBEN SIE DEN NAMEN EIN" TO Sname
```

(und s. Abschnitt 4.1.)

APPEND APPEND

a) APPEND FROM <dateiname> [FOR<bedingung>][SDF][DELIMITED]
b) APPEND BLANK
c) APPEND

Mit APPEND werden neue Datensaetze an die bereits existierende und durch USE eroeffnete Datei angehaengt.

APPEND ist auch fuer die Faelle besonders geeignet, wo die Struktur der Datei veraendert werden soll, d.h. Felder in ihrer Laenge zu veraendern sind, Felder aus der Satzstruktur entfernt oder neue Felder in die Datei aufgenommen werden sollen.

Mit dem Befehl CREATE wird zu diesem Zweck eine neue Datei mit der gewuenschten Struktur erstellt und an die noch leere Datei wird anschliessend mit APPEND die gesamte alte Datei angehaengt. Felder, die nur in der neuen Datei vorkommen, werden mit Leerzeichen gefuellt.

Zum Anfuegen von Saetzen dienen ausser APPEND nur noch die Befehle CREATE und INSERT, wobei INSERT nur das Einfuegen eines Satzes erlaubt, mit APPEND und CREATE jedoch mehrere Saetze hinzugefuegt werden koennen.

a) Diese Form erlaubt die Entnahme von Saetzen aus der mit <dateiname> spezifizierten Datei, im folgenden als Quelldatei bezeichnet. Fehlen im Befehl die Steuerwoerter SDF oder DELIMITED, wird vorausgesetzt, dass die Quelldatei eine Datenbankdatei im REDABAS-Format ist. Die Strukturen der aktivierten Datei und der Quelldatei werden miteinander verglichen, und es werden die Felder, deren Feldnamen in beiden Dateien erklaert sind, in den anzuhaengenden Satz uebernommen. Dabei gilt: Sind die neu eingegebenen Feldinhalte kuerzer als die Felder der aktiven Datei, werden sie rechts mit Leerzeichen aufgefuellt, sind sie laenger, werden sie rechtsseitig abgeschnitten und so auf das Format der vorhandenen Felder gebracht. Es werden so lange Saetze an die aktivierte Datei angehaengt, bis in der Quelldatei keine Saetze mehr vorhanden sind.

Ist SDF (System Daten Format) angegeben, wird angenommen, dass die Quelldatei das Standard-ASCII-Format hat, d.h. jeder Datensatz ist durch <RETURN> und Zeilenvorschub (hexadezimal '0D0A') abgeschlossen. Das Dateiende ist hexadezimal mit '1A' gekennzeichnet. Wird SDF ohne DELIMITED verwendet, erwartet REDABAS die Felder in der Quelldatei ohne Trennzeichen lueckenlos angeordnet. Die Feldlaengen muessen den definierten Feldlaengen in der Datenbankdatei entsprechen.

Ist DELIMITED angegeben, setzt REDABAS voraus, dass die Datenfelder der Quelldatei durch Kommas voneinander getrennt sind. Viele Programmiersprachen erzeugen Dateien mit durch Kommas getrennten Feldern und in Hochkommas oder Anfuehrungszeichen eingeschlossenen Zeichenreihen. REDABAS loescht diese Sonderzeichen und fuegt die Daten im REDABAS-Format an die aktivierte Datei an. APPEND akzeptiert als Trennzeichen nur das Komma und als Begrenzungzeichen ausschliesslich Hochkomma oder Anfuehrungszeichen. Bei Verwendung des COPY-Befehls ist dies zu beachten.

Mit der FOR-Klausel koennen die zu uebernehmenden Saetze selektiert werden. Es wird jeder Satz der Quelldatei ueberprueft, ob er die <bedingung> erfuellt und nur dann wird er angehaengt. Andernfalls wird zum naechsten Satz der Quelldatei uebergegangen, so lange, bis deren Ende erreicht ist.

b) Diese Variante des APPEND-Befehls haengt einen leeren Satz an die aktivierte Datei. Die Felder koennen dann ueber die Befehle EDIT oder REPLACE gefuellt werden.

c) Bei der dritten Form wird die Eingabe des Datensatzes Feld fuer Feld ueber die Erfassungsmaske vorgenommen. Nach Eingabe eines Datensatzes wird die Satznummer erhoeht und die Maske erscheint erneut. Das geschieht so lange, bis anstelle der Dateneingabe fuer das erste Feld die RETURN-Taste gedrueckt wird. Der Anwender muss selbst dafuer Sorge tragen, dass das Fassungsvermoegen der Diskette nicht ueberschritten wird.^

Als Erfassungsschema stellt REDABAS standardmaessig das im CREATE-Befehl definierte Feldschema bereit. Der Nutzer kann jedoch auch eine eigene Erfassungsmaske verwenden. Dazu sind

die gewuenschten @...SAY...GET...-Befehle in einer Maskendatei (.MSK) z.B. mit dem Befehl MODIFY COMMAND zu speichern.

Vor Ausfuehrung des APPEND-Befehls wird diese Erfassungsmaske ueber den Befehl SET FORMAT TO <maskendatei> bereitgestellt. Eigene Erfassungsmasken sind dann vorteilhaft, wenn nicht alle Felder des Satzes erfasst werden sollen, Erlaeuterungen zur Erfassung gewuenscht sind und Vorschriften zum Feldinhalt bestehen.

Ausser bei Form b) gilt, dass bei einer indizierten Datei die im Befehl USE angegebenen Indexdateien automatisch aktualisiert werden, sobald durch APPEND ein neuer Datensatz hinzugefuegt wird. Jede nicht im USE-Befehl aufgefuehrte Indexdatei der eroeffneten Datei muss nach einem APPEND-Befehl neu indiziert werden.

Im Seitenmodus bewirkt der dem APPEND-Befehl vorausgegangene Befehl SET CARRY ON (vgl. Befehl SET) die Uebernahme aller Daten aus dem vorangegangenen Datensatz in den neuen, um anschliessend Aenderungen vornehmen zu koennen. Enthalten aufeinanderfolgende Saetze viele gleiche Daten, erweist sich dies als besonders nuetzlich. In der Form b) werden keine Daten vom letzten Datensatz uebernommen.

Beispiel:

APPEND FROM kunden FOR kunr > 1020

(und s. Abschnitt 3.1.2.)

BROWSE BROWSE

BROWSE [FIELD <feldnamen>]

Der Befehl BROWSE gehoert zu den wirkungsvollsten Befehlen fuer die Anzeige und Aenderung von Daten. Es koennen bis max. 19 Datensaetze einer Datei gleichzeitig auf dem Bildschirm angezeigt werden (in Abhaengigkeit von der Bildschirmgroesse und der Feldlaenge). Dabei werden die Datensaetze zeilenweise dargestellt, und in jeder Zeile erscheinen so viele Felder, wie diese fassen kann. Dieser Ausschnitt der Datei wirkt wie ein Fenster, das man durch die Tastenkombination <CONTROL-B> und <CONTROL-Z> verschieben kann. Gibt es Felder mit mehr Zeichen als auf einer Bildschirmzeile Platz finden, wird der Rest des Feldinhaltes auf der folgenden Zeile fortgesetzt und es verringert sich die Anzahl der angezeigten Saetze entsprechend.

Mit der FIELD-Klausel ist es moeglich, nur die unter <feldnamen> spezifizierten Datenfelder anzeigen zu lassen.

BROWSE beginnt mit dem Datensatz, auf den der Datensatzzeiger zeigt. Will man sich eine Datei vom ersten Datensatz an anzeigen lassen, muss der Datensatzzeiger z.B. mit GO TOP auf den Dateianfang gestellt werden.

REDABAS arbeitet bei Anwendung des Befehls BROWSE im Seitenmodus. Das heisst, der Cursor kann in beliebiger Richtung auf jeden beliebigen Satz, jedes beliebige Feld oder jedes beliebige Zeichen bewegt werden, um Aenderungen vorzunehmen. In der linken oberen Ecke des Bildschirms ist die Nummer des Datensatzes angegeben, auf dem sich der Cursor gerade befindet.

Ausser den Cursortasten koennen mit BROWSE folgende Tastenkombinationen verwendet werden:

CONTROL-G	Loeschen des durch den Cursor positionierten Zeichens
CONTROL-X CONTROL-F	Bewegen des Cursors zum naechsten Datenfeld
CONTROL-E CONTROL-A	Bewegen des Cursors zurueck zum vorhergehenden Feld
CONTROL-D	Bewegen des Cursors um ein Zeichen vorwaerts
CONTROL-S	Bewegen des Cursors um ein Zeichen zurueck
CONTROL-W	Speichern des aktuellen Datensatzes auf die Diskette und Beenden des Seitenmodus
CONTROL-C	Bewegen des Cursors durch alle Felder mit gleichem Namen senkrecht nach unten
CONTROL-R	Bewegen des Cursors durch alle Felder mit gleichem Namen senkrecht nach oben
CONTROL-U (Ver.1.0) CONTROL-L (Ver.2.0)	Ein- / Ausschalten der Loeschmarkierung fuer einen Datensatz
CONTROL-Q	Abbruch des Seitenmodus und Ueberleiten zum REDABAS-Befehlsmodus, wobei die durchgefuehrten Aenderungen unwirksam werden, d.h. es folgt kein Speichern
CONTROL-B	Verschieben des "Fensters" um ein Datenfeld vorwaerts
CONTROL-Z (Ver.1.0) CONTROL-O (Ver.2.0)	Verschieben des "Fensters" um ein Datenfeld zurueck

Beispiel:

```
BROWSE FIELD name, vorname, ort
```

(und s. Abschnitt 3.3.)

CALL [<speichervariable>]

Mit dem CALL-Befehl kann ein in Maschinensprache geschriebenes Unterprogramm aufgerufen werden. Die Anfangsadresse dieses Programms ist in einem vorausgegangenen SET CALL TO <adresse> Befehl zu nennen. CALL verzweigt zu dieser Adresse. Fuer eine korrekte Rueckkehr zu REDABAS ueber den RET-Befehl der Maschinensprache hat der Nutzer Sorge zu tragen.

Wird eine <speichervariable> im CALL-Befehl genannt, so uebergibt REDABAS die Anfangsadresse der <speichervariablen> an das Unterprog- | Adresse der |

	adresse	adresse	Speichervariablen
SCP	Registerpaar DE	Stack	Registerpaar HL
SCP1700	Register CX	Stack	Register BX

Das Unterprogramm darf die Feldlaenge der <speichervariablen> nicht veraendern. Je nach Feldtyp ist die interne Darstellung der Speichervariablen folgendermassen:

Feldtyp	Laenge	Inhalt
logisch	1	FFH : ja 00H : nein
alpha-numerisch	variabel	Byte 0 - Laenge der Zeichenreihe (hex.) Folgebytes - Inhalt der Zeichenreihe
numerisch	6	Byte 0: Bit 0 = 0 : positive Zahl = 1 : negative Zahl 1 = 0 : Zahl (absolut) < 1 = 1 : Zahl (absolut) > 1 2-7 = n : (Ziffernposition) Bit 1 = 0 : 63-n = Position der 1.Ziffer nach dem Dezimalpunkt (mit 0 beginnend) Bit 1 = 1 : n = Position der 1.Ziffer vor dem Dezimalpunkt (mit 0 beginnend) Byte 1 - 5 : 10 Ziffern (je ein Halbbyte), beginnend mit der 1.Ziffer # 0 Beispiel: Zahl: interne Darstellung (hex.) 123.45 42,12,34,50,00,00 -0.052 BE,52,00,00,00,00

CANCEL

Mit CANCEL kann man einen Programmablauf beenden. Die Ausfuehrung einer Befehlsdatei wird abgebrochen, und der REDABAS-Befehlsmodus wird reaktiviert. Die Verwendung dieses Befehls ist nur innerhalb von Befehlsdateien sinnvoll.

Beispiel:

```
ACCEPT 'MENUE BEENDEN? (J/N)' TO Sende
IF !(Sende) = 'J'
    CANCEL
ENDIF
```

CHANGE

CHANGE [<bereich>] FIELD <feldnamen> [FOR <bedingung>]

CHANGE ermoeglicht dem Nutzer problemlos das Aendern einzelner oder auch mehrerer Datenfelder. Es wird jedes eventuell zu aendernde Datenfeld in der unter <feldnamen> spezifizierten Reihenfolge angezeigt. Der Nutzer kann vorhandene Daten aendern, neue Daten eingeben oder auch ohne den Feldinhalt zu aendern, zum naechsten Feld uebergehen. Wurden alle unter <feldnamen> angegebenen Datenfelder angezeigt, wird mit der Anzeige des ersten Feldes des durch <bereich> spezifizierten naechsten Satzes fortgefahren. Die Voreinstellung fuer <bereich> ist nur der jeweils aktuelle Satz.

Mit der FOR-Klausel koennen die Datenfelder entsprechend der <bedingung> selektiert werden.

Alphanumerische Felder koennen zeichenweise geaendert werden, indem auf die Frage "AENDERN VON?" der zu aendernde Feldteil und nach der Aufforderung "AENDERN AUF:" der neue Feldteil eingetippt werden.

Ein <RETURN> auf die 1. Frage vermeidet eine weitere Aenderung. Um ein Feld zu loeschen, ist auf beide Ausschriften mit <CONTROL-Y> und <RETURN> zu antworten. Numerische Felder werden stets vollstaendig geaendert.

Will man den CHANGE-Befehl abbrechen, wird die ESCAPE-Taste gedrueckt.

Beispiel:

```
CHANGE FIELD name

CHANGE ALL FIELD datum FOR ort='dresden'
```

(und s. Abschnitt 3.1.3.)

CLEAR CLEAR

CLEAR [GETS]

CLEAR mit dem Steuerwort GETS entaktiviert alle durch @-Befehle aktivierten GET-Klauseln. Im Unterschied zum Befehl ERASE bleibt der Bildschirm unveraendert.

Ohne GETS versetzt der Befehl CLEAR REDABAS in den Anfangszustand. Das heisst, alle Dateien werden geschlossen (ausser Maskendateien) und alle Speichervariablen geloescht. Ausserdem stellt CLEAR den Arbeitsbereich PRIMARY ein (vergl. Befehl SELECT).

Beispiel: s. Abschnitt 4.1.

CONTINUE CONTINUE

CONTINUE

Dieser Befehl veranlasst nach einem vorangegangenem LOCATE-Befehl die weitere Suche nach dem naechsten Datensatz. Zwischen LOCATE und CONTINUE koennen auch andere Befehle ausgefuehrt werden.

Weitere Zusammenhaenge zwischen LOCATE und CONTINUE siehe Befehl LOCATE

Beispiel: s. Abschnitt 3.2.2.

COPY COPY

COPY TO <dateiname> [<bereich>] [FIELD <feldnamen>]
[FOR <bedingung>] [SDF] [STRUCTURE [EXTENDED]] [DELIMITED [WITH <begrenzung>]]

COPY kopiert die mit USE eroeffnete Datei in die durch <dateiname> spezifizierte Datei, die im folgenden als Zieldatei bezeichnet wird. Liegt die Zieldatei vor, muss sie im REDABAS-Format oder, wenn die Option SDF (System Daten Format) angegeben ist, im Standard-ASCII-Format sein. Ist die Zieldatei noch nicht vorhanden, wird sie angelegt. Voreinstellung fuer <bereich> ist ALL.

Wird das Steuerwort STRUCTURE angegeben, wird eine Datenbankdatei ohne Datensaetze angelegt, d.h. es wird nur die Struktur der aktivierten Datei in die Zieldatei uebernommen.

Gibt man die FIELD-Klausel an, kann man den Kopiervorgang auf bestimmte Felder einschraenken, die unter <feldnamen> nacheinander aufgefuehrt werden, fehlt die FIELD-Klausel werden alle Felder in die Zieldatei uebernommen.

Das Gleiche gilt fuer den Befehl COPY...STRUCTURE FIELD <feldnamen>, wo nur die spezifizierten Felder in der Zieldatei angelegt werden.

Das Gleiche gilt fuer den Befehl COPY...STRUCTURE FIELD <feldnamen>, wo nur die spezifizierten Felder in der Zieldatei angelegt werden.

Die Option EXTENDED bei einem Befehl COPY...STRUCTURE bewirkt, dass nur die komplette Struktur der aktivierten Datenbank in Datensaetze der Zieldatei uebertragen wird, die FIELD-Angaben werden ignoriert. Die Strukturbeschreibung der aktivierten Datei wird zum Dateninhalt der Zieldatei und die Struktur der Zieldatei gibt REDABAS wie folgt vor:

```
FELD     NAME        TYP LAENGE   DEZ

001      FELD:NAME    C    010
002      FELD:TYP     C    001
003      FELD:LAEN    N    003
004      FELD:DEZ     N    003
** SUMME **             00018
```

Die Zieldatei enthaelt folglich so viel Datensaetze wie in der Quelldatei Felder definiert wurden.

Damit wird dem Benutzer die Moeglichkeit gegeben, auf die Struktur einer Datei mittels Befehlsdatei zuzugreifen, um z.B. den weiteren Programmablauf von der Struktur der gerade aktivierten Datei abhaengig zu machen. Auf diese Weise kann man auch die Struktur einer neuen Datei modifizieren (vgl. Befehl CREATE).

Ist SDF im COPY-Befehl angegeben, legt REDABAS eine Datei im Standard-ASCII-Format an, damit die erstellte Datei auch mit anderen Programmen als REDABAS bearbeitet werden kann (z.B. mit einem Textverarbeitungssystem). Jeder Datensatz in der Zieldatei enthaelt dann als Satzendezeichen hexadezimal "0D0A" (Wagenruecklauf ,Zeilenschaltung).

SDF ohne das Steuerwort DELIMITED erzeugt eine .TXT Datei, in der die Felder lueckenlos, in der definierten Laenge ohne Trenn- und Begrenzungszeichen angeordnet sind.
Die nur fuer die Arbeit mit REDABAS benoetigte Struktur der aktivierten Datei wird dabei nicht in die Zieldatei uebernommen. Das bedeutet, dass sich STRUCTURE und SDF-Klauseln gegenseitig ausschliessen.

Mit dem Steuerwort DELIMITED wird veranlasst, dass alle Felder in der Zieldatei durch Kommas voneinander getrennt werden und die Inhalte der Felder vom Typ Zeichenreihe in Hochkommas eingeschlossen werden.

Mit der WITH-Klausel ist es moeglich, ein anderes Sonderzeichen zu verwenden, das die Zeichenreihen einschliesst. Nicht belegte Stellen am Ende von alphanumerischen Feldern werden entfernt. Bei Verwendung des Kommas als Trennzeichen werden auch bei numerischen Feldern fuehrende Leerstellen entfernt. Die Angabe eines Kommas in der WITH-Klausel hat die gleiche Wirkung wie der COPY-Befehl ohne WITH-Klausel, d.h. es werden nur Kommas als Trennzeichen gesetzt, aber keine Begrenzungszeichen ausgegeben. DELIMITED schliesst SDF automatisch ein.

Wenn man eine Zieldatei auf diese Weise erstellt, um sie spaeter mit APPEND wieder einzulesen, so ist zu beachten, dass APPEND nur Hochkommas und Anfuehrungszeichen als Feldbegrenzung akzeptiert.

Enthaelt der <dateiname> keine explizite Angabe des Dateityps, ergaenzt REDABAS automatisch ".DBD" bzw. wenn DELIMITED oder SDF im Befehl angegeben ist ".TXT".

Mit der FOR-Klausel kann fuer ein selektives Kopieren eine <bedingung> gesetzt werden.

Datensaetze, die zur Loeschung markiert sind, werden von COPY nicht verarbeitet. Man kann damit COPY zur physischen Loeschung von Saetzen verwenden.

Beispiel:

COPY TO bestellung DELIMITED

(und s. Abschnitt 3.3.)

COUNT [<bereich>][FOR<bedingung>][TO<speichervariable>]

Mit diesem Befehl kann man die Anzahl der Datensaetze der aktivierten Datei innerhalb des <bereiches> zaehlen. Voreinstellung ist ALL. Bei Angabe einer FOR-Klausel werden nur die Datensaetze gezaehlt, die die <bedingung> erfuellen.

Mit der Klausel TO wird das Ergebnis der Zaehlung als ganzzahliger Wert in der <speichervariablen> abgelegt.

REDABAS gibt das Ergebnis der Zaehlung als Nachricht aus:
"ZAEHLER = xxxxx"

Beispiel:

COUNT FOR kosten > 0 TO Srechn

(und s. Abschnitt 3.5.2.)

CREATE CREATE

a) CREATE [<dateiname>]
b) CREATE <dateiname> FROM <dateiname1>

Mit diesem Befehl wird eine neue Datenbankdatei (Dateityp.DBD) erzeugt. Der Nutzer kann den Dateinamen, die Feldbezeichnungen und die Struktur der Datei frei bestimmen.

a) Wurde kein <dateiname> spezifiziert, fordert REDABAS den Nutzer zur Eingabe desselben auf. Es erscheint die Nachricht: GEBEN SIE DEN DATEINAMEN EIN:

Fuer den Dateinamen gilt ausser den SCP-Konventionen die Einschraenkung, dass Sonderzeichen nur dort zugelassen sind, wo sie bei SCP eine besondere Bedeutung haben (beispielsweise der Doppelpunkt in "B:" fuer die Bezeichnung von Laufwerk B). Der Nutzer muss darauf achten, dass pro Diskette jeder Dateiname nur einmal vergeben wird. Eine bereits unter <dateiname> existierende Datei wird zerstoert.

Nach Eingabe des CREATE-Befehls wird der Nutzer durch folgende Ausschrift zur Definition der Struktur der neuen Datei aufgefordert:

GEBEN SIE DIE SATZSTRUKTUR WIE FOLGT EIN:
FELD NAME,TYP,LAENGE,ANZAHL DEZIMALSTELLEN
001

Der Nutzer muss fuer jedes Feld der neuen Datei die obigen Angaben machen. Der NAME kann max. 10 Zeichen lang sein, muss mit einem Buchstaben beginnen und setzt sich wahlweise aus weiteren Buchstaben, Ziffern und Doppelpunkten zusammen.

Der TYP kann entweder sein:
C fuer Zeichenreihen (alphanumerisch)
N fuer numerische Werte

L fuer logische Werte (einstellig:J,j und T,t fuer wahr
N,n und F,f fuer falsch)

Die LAENGE wird bei alphanumerischen Feldern durch den laengsten aufzunehmenden Wert bestimmt, bei numerischen Werten muss zwischen ganzen Zahlen und Zahlen mit Dezimalpunkt unterschieden werden. Bei ganzen Zahlen ist die Feldlaenge gleich der Anzahl der Ziffern der groessten im Feld zu speichernden Zahl. Zahlen mit Dezimalpunkt erfordern zwei Laengenangaben: die maximale Anzahl der Stellen einschliesslich Dezimalpunkt (unter LAENGE einzugeben) und die Anzahl der Stellen hinter dem Dezimalpunkt (unter ANZAHL DEZIMALSTELLEN einzutragen). Wird ein Vorzeichen gewuenscht, muss eine zusaetzliche Stelle beruecksichtigt werden. Logische Werte haben immer die Laenge 1.

Soll die Eingabe der Dateistruktur abgeschlossen werden, wird anstelle der Eingabe eines neuen Feldnamens die RETURN-Taste gedrueckt. Wird auf die anschliessende Frage
WOLLEN SIE JETZT DATEN EINGEBEN (J/N)?
mit N (fuer Nein) geantwortet, ist der CREATE-Befehl beendet. Auf die Antwort J (fuer Ja) werden die soeben definierten Felder oder eine eigene, vor CREATE definierte Maske auf dem Bildschirm aufgelistet, und der Nutzer kann sie mit dem beabsichtigten Inhalt fuellen. Eine eigene Erfassungsmaske wird mit @...SAY...GET-Befehlen erzeugt und vor Ausfuehrung des CREATE-Befehls durch den Befehl SET FORMAT TO <maskendatei> bereitgestellt. (vgl. Befehl APPEND) Die Dateneingabe wird mit <RETURN> abgeschlossen, vorausgesetzt der Cursor steht am Anfang des ersten Feldes eines neuen Satzes.

b) Wenn der Befehl CREATE eine FROM-Klausel enthaelt, wird die Struktur der neuen Datei den Datensaetzen der unter <dateiname1> spezifizierten Quelldatei entnommen.

Enthaelt die Quelldatei Datensaetze, dann setzt REDABAS voraus, dass diese Datei mit dem Befehl COPY STRUCTURE EXTENDED erzeugt worden ist und entnimmt die Struktur fuer die neue Datei den Datensaetzen der Quelldatei.

In der Form b) des CREATE-Befehls wird keine Dateneingabe angefordert. Datensaetze koennen anschliessend durch APPEND eingegeben werden.

Beispiel:

CREATE rechnung

(und s. Abschnitt 3.1.1.)

a) DELETE [<bereich>][FOR<bedingung>]
b) DELETE FILE <dateiname>

DELETE markiert die im <bereich> angegebenen Datensaetze, die gegebenfalls noch durch die FOR-Bedingung selektiert werden koennen, fuer eine spaetere Loeschung. Die Voreinstellung fuer den <bereich> ist nur der aktuelle Satz. Physisch werden die Saetze erst durch den Befehl PACK geloescht. Fuer die Befehle COPY, APPEND, REPLACE und SORT gelten sie jedoch als geloescht und werden daher durch diese Befehle nicht mehr beruecksichtigt.

Mit RECALL koennen zur Loeschung gekennzeichnete Saetze reaktiviert werden. Die Loeschmarkierung wird zwischen der Satznummer und dem ersten Feld in Form eines Sterns "*" vorgenommen, das Anzeigen der auf diese Weise markierten Saetze ist moeglich.

Form b) erlaubt das Loeschen einer mit <dateiname> bezeichneten Datei. Voreinstellung des Dateityps ist .DBD. Im Gegensatz zur Loeschung der oben beschriebenen einzelnen Datensaetze wird die genannte Datei sofort physisch geloescht.

Fuer das gleichzeitige Loeschen mehrerer Dateien kann das Ersetzungszeichen "?" im <dateinamen> verwendet werden. Das Zeichen "*" ist hier nicht zugelassen.

Eine mit USE aktivierte Datei kann nicht geloescht werden.

Beispiel:

```
DELETE RECORD 5

DELETE NEXT 7

DELETE ALL FOR kosten < 300
```

(und s. Abschnitt 3.1.4.)

DISPLAY DISPLAY

a) DISPLAY [<bereich>][FOR <bedingung>][<ausdrucksfolge>][OFF]
b) DISPLAY STRUCTURE
c) DISPLAY MEMORY
d) DISPLAY FILES [ON <laufwerk>][LIKE <muster>]
e) DISPLAY STATUS

DISPLAY ist einer der wichtigsten Befehle von REDABAS, denn die Anzeige von Daten in beliebigen Kombinationen und Formen gehoert zu den Schwerpunkten jeglicher Datenbankarbeit.

Form a) ermoeglicht die Anzeige einer durch USE eroeffneten Datei. DISPLAY ohne Angabe von <bereich> und <bedingung> bezieht sich nur auf den aktuellen Satz. Ist eine FOR-Klausel vorhanden, lautet die Voreinstellung fuer <bereich> ALL, d.h. es werden all die Saetze angezeigt, die die <bedingung>

erfuellen.

Mit <ausdrucksfolge> werden die Felder spezifiziert, deren Feldinhalte zur Anzeige gebracht werden sollen. Am Anfang jeder Ausgabezeile erscheint dabei die Datensatznummer. Gibt man das Steuerwort OFF an, kann man die Ausgabe der Satznummer unterdruecken.

Nach jeweils 15 Datensaetzen wird die Anzeige nach einem DISPLAY-Befehl unterbrochen. Der Nutzer erhaelt damit Gelegenheit, die angezeigten Saetze zu betrachten. Die Anzeige der naechsten 15 Saetze laesst sich mit einem beliebigen Tastendruck wieder fortsetzen. Dies wird so lange wiederholt, bis alle spezifizierten Datensaetze angezeigt sind. Diese Art der Anzeige mit Pausen nach jeweils 15 Datensaetzen unterscheidet die Befehle LIST und DISPLAY voneinander. LIST zeigt die in Frage kommenden Datensaetze ohne Unterbrechung an. Gemeinsam ist beiden Befehlen, dass sie sich jederzeit durch Betaetigen der ESCAPE-Taste abbrechen lassen.

Wird bei der Anzeige von 15 Datensaetzen die Bildschirmgroesse ueberschritten, kann der Befehl z.B. auch in der Form DISPLAY NEXT 5 eingegeben werden. Mit der Tastenkombination CONTROL-R ist der Befehl mehrfach wiederholbar.

Form b) liefert eine Anzeige der Struktur der eroeffneten Datei.

Form c) listet alle temporaeren Speichervariablen mit deren Inhalt, Groesse und Typ auf.

Form d) bringt alle auf dem voreingestellten oder mit <laufwerk> spezifizierten Laufwerk vorhandenen Dateien vom Typ .DBD gemeinsam mit statistischen Angaben ueber die jeweilige Datei zur Anzeige. Mit LIKE kann man auch Dateien anderen Typs anzeigen lassen, wobei in dem <muster> das Spektrum der anzuzeigenden Dateien gemaess der SCP-Konvention mit "*" und "?" beschrieben werden kann. Das Format dieser Anzeige entspricht dem eines analogen DIR-Befehls in SCP.

Mit Form e) ist es moeglich, aktive Datenbank- und Indexdateien sowie die SET-Parameter aufzulisten.

Beispiel:

```
DISPLAY ALL name,vorname,ort
```

(und s. Abschnitt 3.2.3.)

```
a) DO <befehlsdatei>

b) DO WHILE <bedingung>
        <befehle>
   ENDDO

c)DO CASE
     CASE <bedingung>
           <befehle>
     CASE <bedingung>
           <befehle>
                :
                :
     [OTHERWISE
           <befehle>]
  ENDCASE
```

Form a) des Befehls DO bewirkt die Ausfuehrung der angegebenen <befehlsdatei> (Dateityp .PRG). Die in der Datei gespeicherten Befehle werden so abgearbeitet, als waeren sie ueber die Tastatur eingegeben. Aus einer Befehlsdatei koennen -wiederum ueber DO-Befehle der Form a)- weitere Befehlsdateien aufgerufen werden. Diese DO-Befehle koennen max. 16 mal verschachtelt sein. Der Befehl DO endet bei Dateiende oder durch einen Befehl RETURN. Wurde die Befehlsdatei von einer anderen Befehlsdatei aufgerufen, wird diese weitergefuehrt. Mit CANCEL werden, unabhaengig in welcher Schachtelungsebene dieser Befehl gegeben wurde, alle aktiven Befehlsdateien abgebrochen, und es wird der REDABAS-Befehlsmodus reaktiviert. Eine <befehlsdatei> kann automatisch beim REDABAS-Start aufgerufen werden, wenn REDABAS <befehlsdatei> eingegeben wird.

Beispiel:

```
DO programm
```

(und s. Abschnitt 4.1.)

Form b) des Befehls DO wird in Befehlsdateien zum Initiieren einer Programmschleife verwendet. Die <befehle> der Programmschleife werden so lange durchlaufen, wie die <bedingung> erfuellt ist. Wird die Programmschleife bei Nichterfuellung der <bedingung> verlassen, wird das Programm mit dem auf ENDDO folgenden Befehl fortgesetzt.

Eine DO WHILE - ENDDO -Konstruktion ist im Sinne von REDABAS als ein Befehl zu betrachten. Daraus folgt, dass aehnliche - ebenfalls aus Einleitungs- und Schlussteil bestehende Befehle (z.B. IF - ENDIF) - voll in die DO WHILE-Schleife eingebettet sein muessen.

Beispiel:

```
DO WHILE antwort = J
   STORE name TO Sname
           :
           :
```

```
   SKIP
ENDDO
```

(und s. Abschnitt 4.2.)

Form c) des DO-Befehls wird in Befehlsdateien zur Steuerung einer fallgerechten Verarbeitung von Daten angewandt. Mit DO CASE wird eine bedingte Programmverzweigung eingeleitet, mit ENDCASE wird sie abgeschlossen. Die Anzahl der CASE-Klauseln pro Befehl ist nicht begrenzt. Jeder Programmzweig beginnt mit CASE <bedingung>.

Ist die <bedingung> erfuellt, werden die folgenden Befehle bis zum naechsten CASE <bedingung> ausgefuehrt, und es wird zum Steuerwort ENDCASE gesprungen. Das heisst, dass auch bei mehreren zutreffenden <bedingungen> immer nur die CASE-Klausel der ersten erfuellten <bedingung> ausgefuehrt wird.

Ist die <bedingung> nicht erfuellt, wird die naechste CASE-Klausel ausgewertet.

Sind alle CASE-Klauseln nacheinander ausgewertet worden und war keine <bedingung> erfuellt, bleibt der Befehl ohne Wirkung, wenn keine OTHERWISE-Klausel vorhanden ist. Ist der Zusatz OTHERWISE im Befehl enthalten, werden dessen Befehle ausgefuehrt.

Nach Abarbeitung der durch CASE oder OTHERWISE ausgewaehlten Befehle erfolgt die Programmfortsetzung bei dem Befehl, der auf ENDCASE folgt.

Befehle zwischen DO CASE und der ersten CASE-Klausel werden nicht ausgefuehrt.

CASE-Klauseln duerfen innerhalb einer Befehlsdatei nicht ineinander geschachtelt werden (im Gegensatz zu IF-Klauseln). Wird jedoch aus einer CASE-Klausel in ein Unterprogramm verzweigt, darf dieses Unterprogramm erneut CASE-Klauseln enthalten.

Beispiel:

```
DO CASE
   CASE tag = 'Montag'
     (Befehle fuer Ruhetag)
   CASE tag = 'Sonntag'
     (Befehle fuer Wochenende)
   OTHERWISE
     (Befehle fuer andere Oeffnungstage)
ENDCASE
```

(und s. Abschnitt 4.3.)

EDIT [<zahl>]

Mit EDIT wird der durch <zahl> spezifizierte Datensatz editiert. Der Nutzer ist damit in der Lage, gezielt Aenderungen am Inhalt einzelner Datenfelder vorzunehmen.

Die Anwendung und Wirkungsweise von EDIT ist davon abhaengig, ob REDABAS im Seitenmodus arbeitet oder nicht (vgl. Befehl SET SCREEN).

Im Seitenmodus wird nach Eingabe von EDIT <zahl> die Maske und der Inhalt des mit <zahl> bezeichneten Datensatzes angezeigt. Fehlt die Angabe von <zahl>, wird der Nutzer durch die Nachricht
"GEBEN SIE DIE DATENSATZ-NR. EIN:"
zur entsprechenden Eingabe aufgefordert.
Der Nutzer hat die Moeglichkeit, vor EDIT mit SET FORMAT TO <maskendatei> eine eigene Maske bereitzustellen, um sich nur die fuer ihn im Moment wesentlichen Datenfelder anzeigen zu lassen und gegebenfalls aendern zu koennen (vgl. Befehl APPEND).

Fuer die Aenderung koennen alle die im Abschnitt 3.1.3. unter "Aendern von Datensaetzen" beschriebenen Tastenkombinationen genutzt werden.

Arbeitet REDABAS nicht im Seitenmodus, erscheinen nach Eingabe des Befehls EDIT folgende Ausschriften:
GEBEN SIE EIN: SATZNR., FELD (NR. ODER NAME), NEUER FELDWERT
KOORD:
Wird der neue Feldwert in dieser Eingabe nicht mit angegeben, zeigt REDABAS den bisherigen Wert des Feldes an und fragt bei alphanumerischen Werten
AENDERN VON ? (Es ist der zu ersetzende Teil des Feldes anzugeben)
AENDERN AUF: (Es sind die neuen Werte einzugeben)

Bei numerischen Werten erscheint die Frage
AENDERN AUF: (Der neue Feldinhalt ist komplett einzugeben)
Soll der bisherige Wert belassen bleiben, wird lediglich die RETURN-Taste gedrueckt, ansonsten wird der neue Wert eingegeben. Danach wird mit
KOORD:
wieder nach den Koordinaten des naechsten zu aendernden Feldes gefragt. Aendert sich eine der Koordinaten gegenueber der ersten Eingabe nicht, kann diese bei der folgenden Eingabe ausgelassen werden.

Will man den Befehl EDIT abschliessen, wird als Antwort auf "KOORD:" die RETURN-Taste gedrueckt.

Es ist moeglich, ein Datenfeld vollstaendig zu loeschen, indem man auf die Frage "AENDERN VON ?" bzw. "AENDERN AUF:" <CONTROL-Y> eingibt und <RETURN> drueckt. Danach kann man den neuen Feldinhalt eingeben. Mit <CONTROL-Q> laesst sich der Editiervorgang zu jeder Zeit abbrechen, es wird dann der urspruengliche Inhalt des gerade bearbeiteten Feldes wiederhergestellt.

Wird der Befehl EDIT bei einer indizierten Datei angewendet, wird jeder mit USE aktivierte Schluessel aktualisiert, sobald Aenderungen an den betreffenden Schluesselfeldern vorgenommen werden. Dies gilt selbstverstaendlich nur fuer aktivierte Schluesseldateien.

Beispiel:

EDIT 6

(und s. Abschnitt 3.1.3.)

EJECT EJECT

EJECT

Dieser Befehl veranlasst bei eingeschaltetem und aktiviertem Drucker einen Seitenvorschub, d.h. Positionierung auf den Anfang der naechsten Seite. Vorraussetzung ist die Belegung der SET-Parameter PRINT mit ON und/oder FORMAT mit PRINT (s. Befehl SET). Wird der @-Befehl zur Seitenformatierung verwendet, werden durch EJECT die Zeilen- und Spaltenregister wieder auf Null gesetzt.

ENDDO ENDDO

ENDDO

Mit ENDDO wird das Ende einer DO WHILE-Schleife markiert. Es wird zu dem zugehoerigen DO-Befehl zurueckgesprungen und die <bedingung> erneut ausgewertet (siehe Befehl DO).

ERASE ERASE

ERASE

Dieser Befehl loescht den Bildschirm und positioniert den Cursor in die linke obere Ecke. Wird der @-Befehl zur Bildschirmausgabe verwendet, entaktiviert ERASE alle GET-Klauseln (s. @-Befehl).

Beispiel: s. Abschnitt 4.1.

FIND <zeichenreihe>

Mit FIND ist es moeglich, eine indizierte Datei auf schnelle Weise zu durchsuchen. Es wird der erste Datensatz gefunden, dessen Schluessel mit der angegebenen <zeichenreihe> uebereinstimmt.

Der Befehl FIND laesst sich nur dann anwenden, wenn der Schluessel, nach dem man suchen will, aktueller Hauptschluessel der aktivierten Datei ist (s. Befehl USE). Ist der Schluessel vom Typ Zeichenreihe, genuegt es, als <zeichenreihe> nur die Anfangsbuchstaben des Schluesselwortes anzugeben, nach dem gesucht werden soll. Der FIND-Befehl findet dann den ersten Satz innerhalb der aktivierten Datei, dessen Schluessel entsprechend beginnt.

Mit SET EXACT ON (s. Befehl SET) fuehrt der Befehl FIND nur dann zum Erfolg, wenn Schluesselwert und Wert des Feldes im Datensatz (bis auf Leerzeichen am Feldende) genau uebereinstimmen.

FIND sucht stets vom Dateianfang aus und nur bis zum ersten zutreffenden Satz. Alle weiteren Saetze, auf den der Suchbegriff <zeichenreihe> auch zutrifft, koennen mit FIND nicht gefunden werden. In der Indexdatei selbst folgen sie dem gefundenen Satz unmittelbar.

Wird bei der Sortierung der Index mit Hilfe der !-Funktion gebildet, muss der Suchbegriff ebenfalls aus Grossbuchstaben bestehen, da mit FIND nur die Index-Datei durchsucht wird.

Ist der Schluessel in der Indexdatei numerisch definiert, wird beim Vergleich mit <zeichenreihe> der erste Satz gefunden, dessen Schluessel arithmetisch mit dem Wert von <zeichenreihe> uebereinstimmt.

Der Schluessel in der Indexdatei kann sich auch aus mehreren Feldern zusammensetzen.

<zeichenreihe> kann in Hochkommas oder Anfuehrungszeichen eingeschlossen sein. Am Anfang und Ende muss das gleiche Zeichen stehen. Diese Sonderzeichen sind nur dann erforderlich, wenn der Schluessel, nach dem die Datei indiziert ist, fuehrende Leerzeichen enthaelt. Diese Leerzeichen muessen dann in genauer Anzahl als Bestandteil der <zeichenreihe> innerhalb der Hochkommas bzw. Anfuehrungszeichen vorhanden sein.

Man kann auch den Inhalt einer temporaeren Speichervariablen als Schluesselwert verwenden. Diese Speichervariable ist dann in Form einer Makro-Ersetzung, das heisst mit vorangestelltem "&" anzugeben. Dabei ist eine numerische Speichervariable mit Hilfe der STR-Funktion in einen alphanumerischen Wert umzuwandeln.

Ein mit FIND lokalisierter Satz wird automatisch fuer REDABAS zum aktuellen Satz, d.h. er kann bezueglich seiner Feldinhalte sofort angezeigt, veraendert oder fuer Berechnungen verwendet

werden.

Ist in der Datei kein Satz mit dem angegebenen Suchbegriff vorhanden, erscheint auf dem Bildschirm die Nachricht:
SCHLUESSEL ODER ZEICHENKETTE NICHT VORHANDEN!
Der Datensatzzeiger weist dann auf die Nummer 0. Diese Nummer kann wie jede andere Satznummer mit der Funktion # abgefragt werden.

Man kann den Zugriff auf Datensaetze der indizierten Datei mit SKIP oder LOCATE FOR <bedingung> fortsetzen. SKIP bringt den nach der Ordnung der indizierten Datei folgenden Satz, ohne dabei die Uebereinstimmung des Schluessels zu pruefen. Mit LOCATE arbeitet man, vorausgesetzt der Suchschluessel wird innerhalb der <bedingung> angegeben, nur dann erfolgreich, wenn mindestens ein weiterer Satz mit demselben Schluessel in der Datei vorhanden ist.

Beispiel:

```
FIND 36
FIND &Sbezei
FIND Mueller
```

(und s. Abschnitt 3.4.)

GO oder GOTO GO / GOTO

a) GO RECORD <zahl>
b) GO <zahl>
c) GO TOP
d) GO BOTTOM
e) GO <speichervariable>

Mit GO oder GOTO kann der (REDABAS-interne) Datensatzzeiger auf einen bestimmten Datensatz positioniert werden.

Bei a) und b) wird der Zeiger auf den Datensatz mit der Nummer <zahl> gesetzt (b ist die Kurzschreibweise von a).

Bei c) wird der Zeiger auf den ersten, bei d) auf den letzten Satz der aktivierten Datei gesetzt. Bei indizierten Dateien brauchen die jeweils logisch ersten und letzten Saetze nicht mit den physisch ersten und letzten Datensaetzen uebereinzustimmen, da die logische Reihenfolge von den jeweiligen Werten der Schluessel abhaengt.

Die Variante e) dient zur indirekten Adressierung eines Satzes, die Datensatznummer steht hier in der <speichervariablen>.

Die Einstellung des Datensatzzeigers erfolgt unabhaengig davon, ob der betreffende Datensatz das Loeschkennzeichen traegt.

Beispiel:

```
GOTO RECORD 7

STORE 24 TO Ssatz
GOTO Ssatz
```

(und s. Abschnitt 3.2.4.)

HELP

HELP [<parameter>]

Dieser Befehl macht die ONLINE-Informationen verfuegbar, die in der Datei RDBASMSG.TXT gespeichert sind. Diese Datei muss auf dem aktuellen Laufwerk vorhanden sein.

HELP liefert spezielle Informationen, die durch <parameter> abgerufen werden koennen.

Fuer <parameter> kann eines der folgenden Woerter eingesetzt werden: REDABAS, INSTALL, LEISTUNG, DATEIEN, FUNKTIONEN, BEGRIFFE, TASTATUR, NEU. Ausserdem koennen als <parameter> auch einzelne Befehlswoerter eingegeben werden.

Mit dem Parameter NEU werden Informationen zu erweiterten Leistungen von REDABAS im Vergleich zu einer vorangegangenen Programmversion vermittelt.

IF

```
IF <bedingung>
     befehle>
[ELSE <befehle>]
ENDIF
```

Mit dem Befehl IF koennen von einer <bedingung> abhaengige Programmverzweigungen realisiert werden. Ist die <bedingung> erfuellt, werden die unmittelbar folgenden <befehle> ausgefuehrt. Ist sie nicht erfuellt, werden die dem Steuerwort ELSE folgenden <befehle> ausgefuehrt. ELSE ist wahlfrei. Enthaelt der IF-Befehl keinen ELSE-Teil, und die <bedingung> ist nicht erfuellt, werden alle Befehle bis zum Befehl ENDIF uebergangen. IF und ENDIF bilden ein Klammernpaar, dazwischen koennen weitere IF-Befehle stehen, die Schachtelungstiefe ist nicht begrenzt.

<befehle> sind alle zulaessigen REDABAS-Befehle, dazu zaehlen auch zusammengesetzte Befehle wie IF-ENDIF oder DO WHILE-ENDDO, die als jeweils ein Befehl betrachtet werden. Bei der Verschachtelung solcher Befehlspaare ist auf Korrektheit zu achten. Taucht z.B. in einem der beiden IF-Aeste ein DO WHILE-Befehl auf, muss das zugehoerige Steuerwort ENDDO in demselben Ast, d.h. noch vor dem naechsten ELSE bzw. ENDIF, stehen.

Beispiel:

```
IF ort = 'DRESDEN'
   DO progdre
ELSE
   DO progand
ENDIF
```

(und s. Abschnitt 4.3.)

INDEX INDEX

INDEX ON <ausdruck> TO <indexdatei>

Durch diesen Befehl wird fuer die mit USE eroeffnete Datei (Stammdatei) eine nach dem im <ausdruck> angegebenen Schluessel indizierte Datei angelegt und aktiviert. Diese <indexdatei> enthaelt ausser dem Schluessel, nach dem sie auch sortiert ist, Verweise auf die Stammdatei. In der Stammdatei selbst werden keine Veraenderungen vorgenommen. Es koennen pro Stammdatei beliebig viele <indexdateien> angelegt werden, von denen maximal sieben gleichzeitig aktualisiert werden koennen. Sind mehr als sieben Indexdateien erforderlich, muessen die bei USE noch nicht erfassten Indexdateien erneut indiziert werden.

Durch den INDEX-Befehl sind die Saetze im allgemeinen aufsteigend sortiert, bei numerischen Schluesseln laesst sich jedoch auch absteigend sortieren, vorausgesetzt, der <ausdruck> ist entsprechend formuliert.

Die Indizierung einer Datei ist nur dann zweckmaessig, wenn haeufig (ueber einen bestimmten Schluessel) auf einzelne Datensaetze schnell zugegriffen werden muss (vgl. hierzu Befehl FIND).

Man kann Schluessel auch aus mehreren Feldern kombinieren, um mit FIND eindeutige Ergebnisse zu erhalten (z.B.:INDEX ON name+ort TO iauort).

Eine einmal indizierte Datei kann fuer spaetere Anwendungen entweder in ihrer urspruenglichen Form oder als indizierte Datei bearbeitet werden, da die <indexdatei> nur dann aktiviert wird, wenn sie im Befehl USE explizit mit angegeben ist (s. Befehl USE).

Um die Indizierung lexikalisch korrekt ausfuehren zu koennen, sollte man auf Umlaute moeglichst verzichten, da diese Zeichen im ASCII-Code als Sonderzeichen interpretiert werden und deshalb in der Reihenfolge erst nach den uebrigen Buchstaben kommen. Durch die Verwendung der Grossbuchstabenfunktion wird eine korrekte Sortierung nach Gross- und Kleinbuchstaben erreicht.

Der Index-Befehl bezieht auch zur Loeschung markierte Datensaetze mit ein.

Als Einschraenkung fuer den INDEX-Befehl gilt, dass die TRIM-Funktion nicht Bestandteil von <ausdruck> sein darf und bei Anwendung der ¤- bzw. STR-Funktion die Parameter <start>, <laenge> bzw. <dezimalstellen> nicht mit Hilfe von Variablen oder Ausdruecken dargestellt werden duerfen (vergl. hierzu die entsprechenden Funktionen). Das heisst, es ist beispielsweise anzugeben:

```
INDEX ON ¤('abcdefgh',5,3)+STR(378.456,5,1) TO idatei
```

anstelle von

```
INDEX ON ¤('abcdefgh',x,y)+STR(378.456,lae,dez) TO idatei
```

Im Falle dieser letzten Angabe wird zwar kein Syntax-Fehler durch REDABAS ausgewiesen, aber die Indexdatei waere nicht verwendbar.

Beispiel:

```
USE bestand
INDEX ON bezei TO ibez
```

(und s. Abschnitt 3.4.)

INPUT

INPUT

INPUT ["<text>"] TO <speichervariable>

Mit INPUT koennen vom Nutzer ueber die Tastatur eingegebene Daten in <speichervariablen> abgelegt werden. Dieser Befehl wird in Befehlsdateien verwendet. Wenn die <speichervariable> bis zur Anwendung des Befehls INPUT noch nicht existiert, wird sie erzeugt und der vom Nutzer eingegebene Ausdruck in ihr gespeichert. Wahlweise kann zur Erlaeuterung ein in Hochkommas oder Anfuehrungszeichen (gleiches Zeichen beiderseitig verwenden!) eingeschlossener <text> angegeben werden. Dieser <text> erscheint bei Ausfuehrung von INPUT auf dem Bildschirm und erklaert die geforderte Eingabe.

INPUT kann fuer numerische, logische und alphanumerische Eingaben verwendet werden, wobei letzteres bequemer mit dem ausschliesslich auf alphanumerische Eingabe orientierten Befehl ACCEPT vorgenommen wird, da hier der Ausdruck nicht in Hochkommas oder Anfuehrungszeichen eingeschlossen werden muss.

Der Typ der temporaeren <speichervariablen> wird von REDABAS anhand der Eingabe bestimmt. Numerische Werte werden numerischen Speichervariablen zugewiesen, in Sonderzeichen eingeschlossene alphanumerische Zeichenreihen an alphanumerische Speichervariablen. Bei der Eingabe J, N, T, F (Ja, Nein, Wahr, Falsch) erfolgt die Zuweisung des entsprechenden logischen Wertes an eine logische Variable.

Mit der Funktion TYPE laesst sich der Datentyp der Eingabe bestimmen.

Beispiel:

```
INPUT 'Bitte Betrag eingeben:' TO Srech
```

INSERT [BEFORE] [BLANK]

Mit diesem Befehl kann man einen einzelnen Datensatz in eine Datei einfuegen. Das Einfuegen erfolgt unmittelbar nach dem Datensatz, auf den der Datensatzzeiger gerade zeigt.

Ist die Option BEFORE angegeben, wird der Satz vor dem aktuellen Satz eingefuegt. BLANK bewirkt das Einfuegen eines leeren Satzes. Ist BLANK nicht angegeben, wird der Nutzer zur Eingabe der Feldinhalte aufgefordert.

Wurde vor dem INSERT-Befehl SET CARRY ON (siehe Befehl SET) angegeben, werden die Daten des vorhergehenden Satzes in den neuen Datensatz uebernommen.

Man sollte INSERT nur in dringend notwendigen Faellen anwenden, da dieser Befehl besonders bei grossen nicht indizierten Dateien sehr zeitaufwendig ist.
Bei indizierten Datenbankdateien entspricht INSERT dem APPEND-Befehl, d.h. der neue Satz wird physisch an die Datenbankdatei angehaengt.
Wird der INSERT BLANK-Befehl verwendet, wenn eine Indexdatei aktiv ist, geht REDABAS in den Seitenmodus ueber. Um einen leeren Datensatz in eine indizierte Datei einzufuegen, sollte APPEND BLANK benutzt werden.

Zur Eingabe der Datenfelder wird die Standard-Erfassungsmaske angeboten, wenn nicht der Nutzer zuvor mit dem Befehl SET FORMAT TO <maskendatei> eine eigene Maske bereitgestellt hat (vgl. Befehl APPEND).

JOIN JOIN

JOIN TO <dateiname> FOR <bedingung> [FIELD <feldnamen>]

JOIN zaehlt zu den maechtigsten Befehlen von REDABAS, mit ihm kann man aus zwei Dateien eine dritte erzeugen.

Die beiden zu verknuepfenden Dateien werden im Primaer- und Sekundaerbereich von REDABAS aktiviert (vgl. Befehl SELECT). Vor dem JOIN-Befehl ist der Befehl SELECT PRIMARY zu geben. JOIN positioniert auf den ersten Satz der Primaerdatei und wertet die FOR-Klausel fuer jeden Satz der Sekundaerdatei aus. Bei jedem Satz, der die <bedingung> erfuellt, wird ein neuer Satz in die mit <dateiname> bezeichnete neue Datei geschrieben. Ist das Ende der Sekundaerdatei erreicht, wird in der Primaerdatei auf den naechsten Satz positioniert und die Sekundaerdatei wird wiederum vom ersten Satz an durchlaufen. Dieser Prozess wird bis zum Ende der Primaerdatei wiederholt.

Die Groesse der neuen Datei ist von der Selektionswirkung der FOR-Klausel abhaengig. Maximal kann (bei stets erfuellter <bedingung>) jeder Satz der Primaerdatei mit jedem Satz der Sekundaerdatei verknuepft werden. Das kann zu sehr grossen Dateien fuehren. Es ist zu beachten, dass eine REDABAS-Datei nur max. 65535 Datensaetze enthalten darf.

FIELD ist wahlfrei. Ist es angegeben, erhalten die Saetze der neuen, dritten Datei die Eintragungen der Felder aus der Primaerdatei und gegebenenfalls bis zur Maximalzahl von 32 Feldern die Eintragungen der Sekundaerdatei in der dort angegebenen Reihenfolge. Ist FIELD angegeben, werden nur die unter <feldnamen> spezifizierten Felder in die neue, dritte Datei uebernommen.

Beispiel:

```
USE kunden
SELECT SECONDARY
USE betrag
SELECT PRIMARY
JOIN TO rechnung FOR name = S.name FIELD name,adresse,wert
```

(und s. Abschnitt 3.7.2.)

LIST LIST

a) LIST [<bereich>][FOR <bedingung>][<ausdrucksfolge>][OFF]
b) LIST STRUCTURE
c) LIST MEMORY
d) LIST FILES [ON <laufwerk>] [LIKE <muster>]
e) LIST STATUS

Der Befehl LIST entspricht prinzipiell dem Befehl DISPLAY, wobei folgende Unterschiede zu beachten sind:
- Die Voreinstellung fuer <bereich> ist in jedem Fall ALL.
- Es erfolgt keine Unterbrechung bei der Anzeige nach jeweils 15 Saetzen, die Ausgabe erfolgt fortlaufend, kann aber jederzeit mit der ESCAPE-Taste abgebrochen werden.
- Wird vor dem LIST-Befehl SET DELETE ON eingegeben, werden die zum Loeschen markierten Saetze nicht angezeigt. Wurde SET DELETE OFF eingegeben, werden sowohl bei LIST als auch bei DISPLAY zum Loeschen markierte Saetze angezeigt.

Die Formen b) bis e) von LIST sind also mit denen von DISPLAY vollkommen aequivalent.

Beispiel:

```
LIST name FOR ort = 'DRESDEN'
```

(und s. Abschnitt 3.1.1.)

LOAD LOAD

LOAD <dateiname>

Mit diesem Befehl kann (ab REDABAS Version 3.0) eine vom Nutzer erstellte Assemblerprozedur in den Arbeitsspeicher geladen und mit einem nachfolgenden CALL-Befehl abgearbeitet werden. Die Assemblerprozedur ist eine Datei im ASCII-HEX-Format. Sie kann unter SCP mit dem Programm LINK bzw. unter SCP1700 durch den Assembler ASM86 gebildet werden.

LOCATE [<bereich>][FOR <bedingung>]

Mit LOCATE wird nach dem ersten Datensatz eines <bereiches> gesucht, der die <bedingung> erfuellt. Voreinstellung fuer <bereich> ist ALL. d.h. der Datensatzzeiger steht auf Satznummer 1. Im Gegensatz zum FIND-Befehl sucht LOCATE stets ab der Position in der Datei, auf die der Datensatzzeiger gerade zeigt. Wurde ein Datensatz mit LOCATE gefunden, erscheint auf dem Bildschirm die Nachricht:
SATZ-NR. n

Mit CONTINUE (s. Befehl CONTINUE) kann man die Suche in der Datei fortsetzen, wobei zwischen LOCATE und CONTINUE andere Befehle ausgefuehrt werden koennen. Dadurch wird allerdings die Laenge der <bedingung> auf 128 Zeichen (gegenueber 254) verringert.

Wurde in der aktivierten Datei kein Satz gefunden, der der <bedingung> genuegt, erscheint die Nachricht
DATEIENDE ERREICHT
und der Datensatzzeiger weist auf den letzten Satz der Datei.

Wurde im <bereich> eine NEXT-Klausel verwendet und mit LOCATE wurde kein Datensatz gefunden, erscheint die Nachricht
ENDE DES SUCHBEREICHES
und der Datensatzzeiger zeigt auf den letzten, durch NEXT spezifizierten Satz.

LOCATE arbeitet dann am schnellsten, wenn die Datenbankdatei ohne Indexdatei aktiviert wurde.

Beispiel:

LOCATE FOR gehalt >1000

(und s. Abschnitt 3.2.)

LOOP LOOP

LOOP

Mit LOOP kann man innerhalb einer DO WHILE-Schleife sofort zum Anfang der Schleife zurueckkehren, ohne die Befehle bis zum naechsten ENDDO auszufuehren. Damit kann bei umfangreichen DO WHILE-Schleifen Zeit eingespart werden, da Befehlsfolgen, die nicht bei jedem Durchlauf ausgefuehrt werden muessen, uebersprungen werden koennen. Die Wirkung von LOOP entspricht der von ENDDO, d.h. das Programm wird beim unmittelbar vorangehenden DO WHILE weitergefuehrt.

In der Praxis wird man LOOP nicht unbedingt verwenden, da man z.B. mit IF/ ELSE-Befehlen dieselbe Wirkung, aber wesentlich uebersichtlicher, erreicht.

a) MODIFY STRUCTURE
b) MODIFY COMMAND [<dateiname>]

Form a) dient zum Veraendern der Struktur der aktivierten Datenbank. Es sind alle Aenderungen erlaubt: das Hinzufuegen neuer Felder, das Loeschen vorhandener Felder oder das Veraendern von Feldparametern, wie Name, Typ, Laenge Anzahl Dezimalstellen.

Beim Befehl MODIFY STRUCTURE wird die zu aendernde Dateistruktur angezeigt, und der Nutzer kann mit den bekannten Tastenkombinationen (s. Tabelle Abschnitt 6.2.) Veraenderungen vornehmen.

MODIFY STRhalten, bieten sich mehrere
Varianten an. Zweckmaessig ist, die Datenbankstruktur in eine neue Datei zu kopieren, diese mit USE zu aktivieren und an dieser leeren Datenbankdatei die Strukturaenderung vorzunehmen.

Anschliessend werden mit APPEND die Saetze der alten Datei eingelesen. Danach kann die alte Datei geloescht oder umbenannt werden. und die geaenderte Datei wieder mit ihrem alten Namen versehen werden (s. untenstehendes Beispiel).

Form b) dieses Befehls dient zum Veraendern von Befehls- oder Maskendateien oder auch beliebigen Textdateien. Die Standardeintragung fuer den Dateityp ist .PRG. Wird nur MODIFY COMMAND eingegeben, fordert REDABAS die Eingabe des Dateinamens und -typs. Existiert unter dem eingegebenen Namen noch keine Datei, wird eine neue Datei angelegt und auf dem Bildschirm erscheint die Ausschrift NEUE DATEI. Die neue oder modifizierte Datei wird nach Beendigung des Befehls MODIFY COMMAND unter dem Dateinamen abgespeichert, existiert eine alte Version, erhaelt diese den Dateityp .BAK und bleibt ebenfalls erhalten.

Fuer Form b) sind ebenfalls die fuer Form a) angefuehrten Tastenkombinationen gueltig (vgl. hierzu auch Abschnitt 4.1.).

Der Befehl MODIFY COMMAND unterliegt folgenden Einschraenkungen:

- Dieser Befehl enthaelt im Gegensatz zu den meisten Texteditoren keine Moeglichkeit zur Textsuche oder Blockverschiebung.
- Eine zu aendernde Datei darf nicht groesser als ca. 4000 Bytes sein. REDABAS kann nur bis zu diesem Bereich die Cursorposition kontrollieren.
- Die Laenge einer Zeile darf 77 Zeichen nicht ueberschreiten, einschliesslich nicht sichtbarer Steuerzeichen (wie z.B. Wagenruecklauf/Zeilenvorschub), alle Zeichen ab der 77. Stelle werden abgeschnitten.
- Wird eine Datei auf einem grossen Bildschirm (z.B. 80x24 zeichen erfasst, kann diese nicht auf einem kleinen Bildschirm (z.B. 64x16 Zeichen) geaendert werden, es sei denn, die Zeilenlaenge ist kleiner als 64.

Beispiel:

```
USE bestand
COPY STRUCTURE TO temp
USE temp
MODIFY STRUCTURE
APPEND FROM bestand
DELETE FILE bestand
USE
RENAME temp TO bestand
```

(und s. Abschnitt 3.3. (MODIFY STRUCTURE)
sowie Abschnitt 4.1. (MODIFY COMMAND))

NOTE NOTE

a) NOTE <text>
b) * <text>

Mit NOTE koennen Kommentare in einer Programmdatei untergebracht werden, die beim Abarbeiten des Programmes uebersprungen werden. Im Gegensatz zum Befehl REMARK erscheint der <text> bei der Befehlsausfuehrung nicht auf dem Ausgabemedium.

* und NOTE haben die gleiche Wirkung.

Beispiel:

```
NOTE Programm Rechnung
  *  Programm Kundeninformation
```

(und s. Abschnitt 4.1.)

PACK PACK

PACK

PACK bewirkt die physische Loeschung der durch DELETE mit "*" zur Loeschung markierten Saetze der aktivierten Datei. Nach Ausfuehrung von PACK kann auf diese Saetze nicht mehr zugegriffen werden.

PACK aktualisiert auch alle aktivierten Indexdateien. Weitere Indizes muessen mit dem Befehl INDEX neu erstellt werden. Sehr grosse indizierte Dateien werden jedoch wegen zu grossem Zeitaufwandes zweckmaessigerweise nicht mit PACK aktualisiert. In diesem Fall ist es besser, zunaechst die Datei ohne aktivierten Schluessel mit PACK zu behandeln und anschliessend mit INDEX eine neue Indizierung vorzunehmen.

Nach Ausfuehrung von PACK gibt REDABAS eine Nachricht ueber die Anzahl der in der Datei verbleibenden Saetze aus (s. Beisp.).

Alternativ zu PACK kann COPY zum Loeschen markierter Saetze

verwendet werden. Es werden nur die Saetze (ohne Loeschmarkierung) "kopiert", die alte Datei bleibt als Backup erhalten.

Beispiel:

```
PACK

REDABAS-Nachricht: DATENBANKDATEI VERDICHTET,
                   n SAETZE KOPIERT
```

(und s. Abschnitt 3.1.4.)

POKE

POKE <adresse>, <bytefolge>

Mit dem POKE-Befehl koennen einzelne Bytes im Arbeitsspeicher veraendert werden. Beginnend bei <adresse> ersetzt POKE die Zeichen im Arbeitsspeicher durch die in der <bytefolge> genannten Zeichen. <adresse> und <bytefolge> sind dezimal anzugeben. Dieser Befehl ist mit Sorgfalt zu benutzen, um keine ungewollten Ueberschreibungen zu verursachen.

Beispiel:

```
POKE 42744,34,252,166,201
```

QUIT

QUIT [TO <SCP-befehlsfolge>]

QUIT beendet REDABAS und gibt die Steuerung an das Betriebssystem zurueck. Es werden alle Dateien geschlossen. Auf dem Bildschirm erscheint die Nachricht

```
******************
* REDABAS - ENDE *
******************
```

Mit QUIT TO <SCP-befehlsfolge> kann man REDABAS verlassen und sofort die in der <SCP-befehlsfolge> aufgefuehrten Programme in der dort angegebenen Reihenfolge abarbeiten, ohne erst den Umweg ueber das Betriebssystem nehmen zu muessen. Dabei ist die Anzahl der in der <SCP-befehlsfolge> angegebenen Programmnamen bzw. SCP-Befehle unbegrenzt, es darf jedoch kein Element die Laenge von 127 Zeichen ueberschreiten. Ist das letzte Programm bzw. der letzte Befehl abgearbeitet, wird die Steuerung an SCP uebergeben. Soll anschliessend mit REDABAS weitergearbeitet werden, so kann als letztes Element der <SCP-befehlsfolge> REDABAS angegeben werden.

Die Programmnamen bzw. SCP-Befehle sind in Hochkommas einzuschliessen und durch Kommas voneinander zu trennen.

Bei Benutzung von <SCP-befehlsfolge> wird auf der Diskette im Laufwerk A eine temporaere Datei ¤¤¤.SUB angelegt.

Beispiel:
QUIT TO 'PIP C:=KUNDEN.DBD','DIR C:'

READ READ

READ [NOUPDATE]

Mit READ wird im Seitenmodus die Eingabe und Aenderung des Inhalts von Variablen realisiert, die zuvor mit den Befehlen @...GET aktiviert wurden. Die mit GET <variable> aufgerufenen Datenfelder oder temporaeren Speichervariablen werden an den mit @ <koordinaten> festgelegten Bildschirmpositionen angezeigt und koennen wie bei EDIT geaendert werden, d.h. der Cursor ist frei beweglich. READ speichert die Eingaben bzw. Aenderungen in den betreffenden Datenfeldern bzw. Speichervariablen.
Der Seitenmodus kann zuvor mit SET SCREEN ON (entspricht der Standardeinstellung) aktiviert werden.

Wurde vor dem Befehl READ ein SET FORMAT TO <maskendatei> aufgerufen, werden bei READ die in der <maskendatei> enthaltenen @-Befehle wirksam. Damit kann sich ein Nutzer den Bildschirm je nach Verwendungszweck "massschneidern".

Nach einem Befehl SET FORMAT TO SCREEN kann man den Bildschirm mit ERASE loeschen. Danach kann mit einer Folge von @-Befehlen eine neue Editiermaske erstellt werden, und mit einem folgenden READ-Befehl koennen Daten eingegeben und geaendert werden.

Werden nach diesem READ weitere @-Befehle gegeben, ohne dass der Bildschirm geloescht wurde, so wird der Cursor beim naechsten READ auf die erste Speichervariable gesetzt, die nach dem vorherigen READ angezeigt wurde. So kann die weitere Gestaltung des Bildschirms in Abhaengigkeit von den Eingaben erfolgen, die mit dem ersten READ eingelesen wurden.

Fuer die Cursorbewegung innerhalb und zwischen den Feldern und fuer die Steuerfunktionen sind auch hier die bereits beschriebenen Tastencodes verfuegbar.

Die in den GET-Klauseln des @-Befehls verwendeten Variablen koennen entweder Datenfelder der aktivierten Datei oder temporaere Speichervariablen sein, wobei letztere bereits vor Ausfuehrung des @-Befehls angelegt sein muessen (Gegebenenfalls werden sie mit Leerzeichen entsprechend der erforderlichen Stellenzahl angelegt, wie z.B. STORE ' ' TO Stemp).

READ gibt die GET-Klauseln nicht frei. Werden mehr als 64 GET-Variable benannt (z.B. in einer Schleife), so sind die Variablen jeweils nach READ freizugeben (durch ERASE oder CLEAR GETS).

Mit NOUPDATE wird angezeigt, dass keine indizierten Felder durch READ geaendert werden. Dadurch kann der Indexzugriff entfallen.

Beispiel:

```
@ 4,0 SAY 'Name:' GET name
@ 6,0 SAY 'Adresse:' GET ort
READ
```

(und s. Abschnitt 4.4.)

RECALL RECALL

RECALL [<bereich>][FOR <bedingung>]

RECALL macht die Loeschmarkierung, die durch DELETE gesetzt wurde, rueckgaengig. Damit sind nach Wirksamwerden von RECALL alle Befehle, wie z.B. APPEND, SORT usw. fuer die betreffenden Datensaetze wieder anwendbar. Wenn keine Bereichsangabe erfolgt, wird nur der aktuelle Datensatz reaktiviert.
(vergl. hierzu auch Befehl DELETE).

Beispiel:

```
RECALL RECORD 12
RECALL ALL FOR gehalt > 1000
```

REINDEX REINDEX

REINDEX

Mit diesem Befehl werden alle augenblicklich aktivierten Indexdateien aktualisiert. Normalerweise erfolgt das Aktualisieren automatisch bei Aenderungen in der Datenbankdatei, vorausgesetzt die Indexdatei wurde zuvor im USE-Befehl genannt oder durch SET INDEX ... zugewiesen. REDABAS zeigt den Neuaufbau eines Index durch eine Bildschirmmeldung an.

Beispiel:

```
USE bestand INDEX iortbest
APPEND
:
:
SET INDEX TO iteilbes, ibezbest
REINDEX
```

a) RELEASE [<speichervariablen>]
b) RELEASE ALL
c) RELEASE ALL LIKE <muster>
d) RELEASE ALL EXCEPT <muster>

Form a) dieses Befehls veranlasst das Loeschen der in der <variablenfolge> angegebenen temporaeren Speichervariablen.

Form b) bewirkt die Loeschung aller temporaeren Speichervariablen.

Form c) loescht alle temporaeren Speichervariablen, deren Name dem angegebenen <muster> entspricht.

Form d) loescht alle temporaeren Speichervariablen, ausgenommen die, deren Name dem angegebenen <muster> entspricht.

<muster> kann alle fuer Feldnamen gueltigen Zeichen enthalten, ein "?" kann fuer ein beliebiges Zeichen eingesetzt werden, und ein "*" steht stellvertretend fuer den gesamten Rest des Namens.

Beispiel:

RELEASE Ssum	-	Loeschen einer einzelnen Speichervariablen
RELEASE ALL LIKE S*	-	Loeschen aller Speicher-variablen, deren Name mit "S" beginnt.

REMARK <text>

Mit diesem Befehl wird ein Kommentar in Form eines beliebigen <textes> in ein Programm eingefuegt, der -im Gegensatz zu NOTE- waehrend des Programmlaufes auf dem Ausgabemedium erscheint. Der Befehl REMARK wird in der Regel nur in Befehlsdateien angewandt.

Beispiel:

REMARK Materialuebersicht

RENAME <dateiname1> TO <dateiname2>

Mit RENAME koennen beliebige SCP-Dateien umbenannt werden. Voreinstellung fuer den Dateityp ist ".DBD". Jeder andere Dateityp ist entsprechend zu spezifizieren.

Beispiel:

```
RENAME adressen TO kunden
RENAME prog1.PRG TO prog5.PRG
```

REPLACE REPLACE

```
REPLACE [<bereich>] <feldname1> WITH <ausdruck>
        [,<feldname2> WITH <ausdruck>,...]
        [FOR <bedingung>] [NOUPDATE]
```

REPLACE ersetzt die Inhalte der durch <feldname> spezifizierten Datenfelder einer mit USE aktivierten Datei durch die zugehoerigen <ausdruecke>.

Die Angabe von <bereich> ist wahlfrei, ist <bereich> nicht angegeben, werden die Ersetzungen nur am aktuellen Satz vorgenommen. <ausdruck> kann ein Wert, der Name eines Feldes oder einer Speichervariablen oder eine Formel sein.

Bei Anwendung von REPLACE bei einem Schluesselfeld, das durch die INDEX-Klausel im Befehl USE aktiviert wurde, kann es passieren, dass der aktuelle Satz seine Position in der Datei aendert, und dass der "neue naechste Satz" eventuell nicht mehr mit dem "alten naechsten Satz" uebereinstimmt. Deshalb sollten Schluesselfelder nicht mit REPLACE veraendert werden, wenn fuer <bereich> NEXT n angegeben wird. REPLACE wirkt sich nicht auf Indexdateien aus, die nicht durch USE aktiviert wurden.

Der REPLACE-Befehl wirkt nur auf den durch SELECT zuletzt aktivierten Dateibereich. Fuer den zweiten Dateibereich wird der Befehl zwar angenommen, aber es erfolgt keine bleibende Aenderung des Inhalts. Durch das Loeschkennzeichen markierte Saetze werden durch REPLACE nicht veraendert.

Wird NOUPDATE angegeben, unterbleibt die automatische Aktualisierung aktivierter Indexdateien. Bei zahlreichen Aenderungen ist es vorteilhafter, nach Ausfuehrung der REPLACE-Befehle durch REINDEX die Indexdateien zu aktualisieren. NOUPDATE sollte auch angegeben werden, wenn Indexdateien aktiv sind und der REPLACE-Befehl keine Schluesselfelder aendert.

Beispiel:

```
REPLACE ALL gehalt WITH gehalt * 1.1
```

(und s. Abschnitt 3.1.3. und 3.7.1.)

REPORT [FORM <reportdatei>][<bereich>][FOR <bedingung>]
[TO PRINT] [PLAIN]

Mit dem Befehl REPORT wird der Listengenerator zur Ausgabe einer vorprogrammierten Liste auf dem Bildschirm oder Drucker aufgerufen. REDABAS ermoeglicht dem Nutzer, diese Liste seinen Wuenschen entsprechend im Dialog zu erstellen. Damit wird aufwendiges Programmieren, beispielsweise mit den Ausgabebefehlen ? und @, vermieden.
Mit REPORT wird definiert, welche der Daten der aktivierten Datei auf welche Art aufgelistet werden sollen. Ausserdem sind Spaltenueberschriften, Summenberechnungen ueber numerische Felder und Ausdruecke, deren Werte aus Datenfeldern, Speichervariablen und Konstanten gebildet werden, erlaubt.

Beim Erstellen einer Liste wird die gesamte, vorher mit USE aktivierte, Datei ausgewertet. Mit der FOR-Klausel werden die Datensaetze selektiert, die der <bedingung> genuegen. Ist TO PRINT angegeben, erfolgt die Ausgabe ausser auf dem Bildschirm auch auf dem Drucker. Voreinstellung fuer <bereich> ist ALL, wenn keine anderen Angaben erfolgen.

Die mit REPORT erzeugte Listenstruktur wird in der <reportdatei> abgespeichert, die vom Typ .DEF ist. Fehlt die FORM-Angabe oder ist die <reportdatei> nicht vorhanden, nimmt REDABAS an, dass eine neue Liste definiert werden soll. Wurde die Struktur im Dialog definiert, wird von REDABAS eine Datei unter Erfragung des Namens fuer die <reportdatei> angelegt. Diese Datei kann von spaeter angewandten REPORT-Befehlen wieder verwendet werden, ohne dass die Struktur neu definiert werden muss. Es koennen dann auch neue Festlegungen fuer <bereich> oder die FOR-Klausel getroffen werden.
Beispiele fuer den Dialog zur Erstellung einer Liste zeigen die Bilder 37 und 39 dieser Programmtechnischen Beschreibung (in den Bildern 38 und 40 sind die Listen selbst dargestellt). Wie dort ersichtlich ist, werden Angaben fuer L=linker Rand (Voreinstellung 8 Leerzeichen), Z=Zeilen pro Seite (Voreinstellung 57 Zeilen) und B=Seitenbreite (Voreinstellung 80 Zeichen einschliesslich linker Rand) gefordert. Nach Angaben ueber Seitenueberschrift, zweizeilige Ausgabe, Gesamt- und Zwischensummen (auf die spaeter noch eingegeangen wird) werden fuer jede Spalte des Berichts nacheinander die Anzahl der auszugebenden Stellen (BREITE) und der Name des entsprechenden Feldes (INHALT) erfragt. Die BREITE der Spalte kann unabhaengig von der tatsaechlichen Laenge eines Feldes definiert werden. Ist sie in der Liste mit weniger Stellen angegeben als die Feldlaenge definiert wurde, veranlasst REDABAS eine mehrzeilige Ausgabe.

Ausser Datenfeldern koennen in einer Spalte auch Speichervariablen oder das Ergebnis von Ausdruecken stehen, wie z.B. in Spalte 005 im Bild 39: preis*anz. Die Datei selbst wird dadurch nicht veraendert, da REPORT nur lesend auf die Datei zugreift.

Nach den Angaben fuer die Breite und den Inhalt der Spalten ist die SPALTENBEZEICHNUNG einzugeben. Hier gilt ebenfalls (wie bei allen alphanumerischen Angaben) die Regel, dass mit

einem Semikolon (;) ein Zeilenumbruch ausgeloest wird. Das heisst, der nach dem Semikolon stehende Text kommt auf die naechste Zeile. Eine Anwendung dessen ist z.B. das Unterstreichen einer Textzeile. Mit den Sonderzeichen "<" bzw. ">" kann die Ausgabe der Spaltenueberschrift links- oder rechtsbuendig festgelegt werden, ohne die Angabe eines der Zeichen wird der auszugebende Text in der Spaltenbreite zentriert.

Will man numerische Felder summieren, wird auf die Frage: WUENSCHEN SIE GESAMTSUMMEN? (J/N): mit "J" geantwortet. Dadurch wird bei jeder Spalte mit numerischen Werten diese Frage nochmals gestellt und der Nutzer kann so gezielt Spalten auswaehlen, die jeweils fuer sich zu summieren sind. Das Ergebnis wird unter ** SUMME ** am Ende einer Liste ausgegeben.

Beantwortet man die Frage WUENSCHEN SIE ZWISCHENSUMMEN IN DER LISTE? (J/N): mit "J", erscheint als naechstes die Aufforderung GEBEN SIE DIE FELDNAMEN FUER ZWISCHENSUMMEN AN:. Nach Eingabe des Datenfeldnamens werden alle Saetze, die in diesem Feld den gleichen Wert haben, nacheinander ausgegeben. Sind es numerische Felder, werden sie ueber alle zusammengehoerenden Saetze summiert. Das Ergebnis wird als Zwischensumme ausgegeben. In der Liste werden also Datensaetze gleichen Inhalts im Zwischensummenfeld gruppiert. Der Inhalt dieses Feldes wird am Anfang jeder Gruppe angegeben und kann mit einer Ueberschrift versehen werden: GEBEN SIE DIE UEBERSCHRIFTEN FUER ZWISCHENSUMMEN EIN:. Diese Ueberschrift wird dann von REDABAS mit einem vorangestellten Stern "*" versehen. Nach der Ausgabe der Zwischensumme (mit ** ZWISCHENSUMME ** gekennzeichnet) kann man auf eine neue Seite wechseln:
SEITENVORSCHUB NACH ZWISCHENSUMMEN GEWUENSCHT? (J/N):

Man kann auch auf die Ausgabe der einzelnen Positionen verzichten und nur die Ergebnisse der Summenberechnungen ausgeben. Die entsprechende Frage lautet: LISTE NUR MIT SUMMEN? (J/N):

Der Dialog fuer die Definition der Liste wird beendet, indem die <RETURN>-Taste betaetigt wird, nachdem eine neue Spaltennummer erscheint. REDABAS erstellt daraufhin die Liste und gibt sie auf dem Bildschirm und wahlweise (bei Angabe von TO PRINT) auf dem Drucker aus.

Zusaetzlich zu den oben beschriebenen Moeglichkeiten kann auf die Gestaltung der mit REPORT erstellten Liste durch folgende Befehle Einfluss genommen werden:

- SET DATE TO - Ausgabe des Datums unterhalb der Seitennumerierung
- SET HEADING TO - Moeglichkeit fuer eine weitere Seitenueberschrift
- SET EJECT OFF - Unterdrueckung des Seitenvorschubs (der sonst am Anfang einer Liste kaeme)

(Vgl. weitere Einzelheiten bei den SET-Befehlen)

Diese oben angefuehrten Moeglichkeiten bleiben allerdings nur waehrend einer REDABAS-Sitzung aktiv, im Gegensatz zu den

Angaben der <reportdatei>.

Mit der Option PLAIN wird die Angabe der Seitennummer und des Datums unterdrueckt, die Seitenueberschrift erscheint nur auf der ersten Seite.

Als Hinweis sei noch vermerkt, dass die <reportdatei> mit Texteditoren bearbeitet werden kann (in REDABAS mit MODIFY COMMAND). Ohne dass eine Liste mit REPORT neu definiert werden muss, koennen Aenderungen vorgenommen werden, die allerdings den Moeglichkeiten des REPORT-Befehls nicht widersprechen duerfen, d.h. die Eintragungen in die <reportdatei> muessen den Dialogantworten entsprechen.

In der Regel wird jedoch die Neuerstellung der <reportdatei> im REPORT-Dialog sinnvoller sein.

Beispiel:

```
REPORT FORM persdat FOR sprache = 'englisch' TO PRINT
```

(und s. Abschnitt 3.6.)

RESET [<laufwerk>]

RESET ermoeglicht die Fortsetzung eines REDABAS-Programms nach einem Diskettenwechsel, ohne dass die Steuerung an das Betriebssystem uebergeben werden muss.

Es werden die von Betriebssystem verwalteten Informationen ueber die Disketten in den Laufwerken aktualisiert. Nach Diskettenwechsel darf seitens des Betriebssystems nicht auf die neu eingelegte Diskette geschrieben werden, solange das Betriebssystem noch nicht gestartet ist. Das Starten geschieht entweder ueber die Hardware, d.h. h
Betaetigen der Tastenkombination <CONTROL-C>. RESET uebernimmt diese Funktion.

Mit <laufwerk> wird das Diskettengeraet bezeichnet, auf dem der Diskettenwechsel erfolgt ist. Die REDABAS-Programmdiskette darf nicht ausgetauscht werden.

Mit RESET werden alle Dateien, die vor dem Diskettenwechsel eroeffnet waren, wieder aktiviert. Voraussetzung ist, dass sie auf der aktiven Diskette noch vorhanden sind. Existieren sie nicht mehr, werden sie von REDABAS aus der Liste der aktivierten Dateien gestrichen. Gibt es nach dem Diskettenwechsel eine andere Datei gleichen Namens, benutzt REDABAS diese neue Datei, die aber korrekterweise mit USE eroeffnet werden muesste. Damit solche Irrtuemer ausgeschlossen werden, muessen vor einem Diskettenwechsel die entsprechenden Dateien geschlossen werden. Dazu dient USE ohne Angabe eines Dateinamens, CANCEL schliesst die im Zugriff befindlichen Befehlsdateien.

RESTORE RESTORE

RESTORE FROM <dateiname> [ADDITIVE]

Mit diesem Befehl wird eine Datei gelesen, in die vorher mit dem Befehl SAVE TO <dateiname> temporaere Speichervariable abgespeichert wurden. Es werden mit RESTORE nur die Speicher-variablen erfasst, die mit SAVE TO <dateiname> gerettet wurden.

Sollen bereits vorher erklaerte Speichervariable erhalten bleiben, ist ADDITIVE anzugeben, andernfalls werden sie ueberschrieben.

Wird bei <dateiname> kein Typ angegeben, ergaenzt REDABAS mit ".VAR".

Beispiel:

```
STORE "ABC" TO Salpha
STORE 123 TO Szahl
STORE "VERHEIRATET" TO Sfam
SAVE TO speicher
```

```
STORE "XYZ" TO Salpha
STORE "DRESDEN" TO Sort


DISPLAY MEMORY
SALPHA     (C)    XYZ
SZAHL      (N)    123
SFAM       (C)    VERHEIRATET
SORT       (C)    DRESDEN
** SUMME **  04 TEMP.VARIABLE, 00031 BYTES LANG

RESTORE FROM speicher

DISPLAY MEMORY
SALPHA      (C)  ABC
SZAHL       (N)  123
SFAM        (C)  VERHEIRATET
** SUMME ** 03 TEMP.VARIABLE, 00023 BYTES LANG
```

RETURN RETURN

RETURN

Der Befehl RETURN wird in einer Befehlsdatei zum Beenden des Programms verwendet. Nach diesem Befehl wird in die Ebene zurueckgekehrt, von der aus das Programm gestartet wurde.

Das heisst, wird RETURN in einer Befehlsdatei erreicht, werden die Befehle hinter dem DO-Befehl ausgefuehrt, der zum Aufruf der jetzt mit RETURN beendeten Befehlsdatei diente. Wenn der Aufruf nicht aus einer uebergeordneten Befehlsdatei erfolgte, erscheint auf dem Bildschirm das von REDABAS ausgegebene Bereitschaftszeichen fuer die Befehlseingabe, der Doppelpunkt (:).

Wird das Ende einer Befehlsdatei erreicht, ohne dass der Befehl RETURN gegeben wurde, hat das die gleiche Wirkung wie ein Befehl RETURN.

Beispiel: s. Abschnitt 4.1.

SAVE SAVE

SAVE TO <dateiname> [ALL LIKE <muster>]

Mit SAVE werden entweder alle augenblicklich definierten oder die mit ALL LIKE <muster> ausgewaehlten temporaeren Speichervariablen mit Namen, Typ und Inhalt in der mit <dateiname> bezeichneten Datei gespeichert. Von dort koennen sie spaeter mit RESTORE (s. Befehl RESTORE) wieder zurueckgeladen werden. Im <muster> koennen durch "?" beliebige Zeichen dargestellt werden, "*" kann fuer alle restlichen Zeichen stehen.

Wird bei <dateiname> kein Typ angegeben, wird von REDABAS ".VAR" realisiert.

Beispiel: s. Beispiel des Befehls RESTORE

SELECT SELECT

SELECT [PRIMARY]
[SECONDARY]

REDABAS erlaubt das gleichzeitige Bearbeiten von zwei Datenbankdateien. Eine Datei wird davon zur Primaer-, die andere zur Sekundaerdatei erklaert.

Nach dem Start von REDABAS wird im Primaerbereich gearbeitet, eine Datei wird mit USE aktiviert. Dieser Primaerbereich bleibt so lange aktiv, bis der Befehl SELECT SECONDARY gegeben wird. In diesem Bereich kann jetzt eine zweite Datei eroeffnet werden. Beide Dateien kann man abwechselnd mit SELECT PRIMARY bzw. SELECT SECONDARY bearbeiten.

Jede der beiden Dateien besitzt ihren eigenen Datensatzzeiger, der beim Wechsel von einer Datei zur anderen auf dem jeweiligen Datensatz stehenbleibt und dort fuer einen spaeteren Zugriff bereit ist. Das ermoeglicht die Satz-zu-Satz-Kommunikation zwischen beiden Dateien.

Sind sowohl Primaer- als auch Sekundaerdatei aktiviert, koennen Datenfelder aus beiden Dateien angesprochen werden. Treten dabei Felder mit gleichem Namen auf, muessen sie durch ein vorangestelltes "P." (Primaer) bzw. "S." (Sekundaer) spezifiziert werden.

Befehle, die eine Aenderung der Positionierung innerhalb einer Datei zu Folge haben (wie z.B. GOTO, SKIP, SORT, COPY usw.) und im Geltungsbereich mehrere Saetze umfassen, betreffen nur den gerade eingestellten Bereich (Primaer oder Sekundaer). Mit dem Befehl SET LINKAGE ON kann jedoch veranlasst werden, dass alle Befehle, die mit einem Geltungsbereich arbeiten, die gleiche Positionierung in beiden Dateien erhalten.
REPLACE betrifft nur den gerade ausgewaehlten Bereich, auch DISPLAY zeigt nur die Daten des gerade eingestellten Bereiches.

Die Arbeit mit SELECT schliesst nicht aus, dass waehrend einer Sitzung weitere Dateien bearbeitet werden koennen, die regulaer (nacheinander) mit USE eroeffnet werden. Bei der Uebertragung von Daten oder fuer das Speichern von Zwischenergebnissen koennen dann je nach Notwendigkeit temporaere Speichervariablen oder Dateien angelegt werden.

Beispiel:

```
USE teil
SELECT SECONDARY
USE lieferant
SELECT PRIMARY
JOIN TO teillief FOR teilnr = S.teilnr;
                 FIELDS S.name, S.adresse...
```

(und s. Abschnitt 3.7.1.)

a) SET <parameter1> [ON]
[OFF]
b) SET <parameter2> TO <...>

Mit SET wird die Konfiguration von REDABAS waehrend einer Sitzung (und nur fuer die Dauer dieser Sitzung) spezifiziert. Damit ist dem Nutzer die Moeglichkeit gegeben, das Systemverhalten seinen Wuenschen entsprechend zu beeinflussen. Form a) dieses Befehls schaltet eine Reihe von Betriebsweisen an (ON) oder aus (OFF).

Form b) ermoeglicht die Veraenderung von Standardwerten einiger Funktionen oder das Treffen einer Auswahl aus mehreren Varianten (im Unterschied zu Form a aus mehr als zwei).

Nachfolgend werden die zulaessigen Parameter der Form a) vorgestellt. Die jeweilige Voreinstellung ist durch Unterstreichen von ON oder OFF gekennzeichnet.

Parameter1	Einstellung	Wirkung
ALTERNATE	ON	Protokollieren der Ergebnisse (Bildschirmanzeigen) in eine Datei
	OFF	Beenden der Protokollierung
BELL	ON	Ertoenen eines Signals bei unzulaessigen Daten bzw. Ueberschreiten der Feldgrenze (nur wenn der Rechner dafuer eingerichtet ist)
	OFF	Unterdruecken der akustischen Warnung
CARRY	ON	Uebertragung von Datenwerten im Seitenmodus (bei APPEND) aus dem vorhergehenden Satz automatisch in den naechsten Satz
	OFF	Keine Uebertragung der Datenwerte von einem Satz in den anderen
COLON	ON	Markieren von Feldgrenzen mit Doppelpunkten bei mit GET eizulesenden Datenfeldern bei der Anzeige (Befehl @)
	OFF	Unterdruecken der Doppelpunkte zur Feldbegrenzung
CONFIRM	ON	Einschalten eines Zwanges, bei der Datenerfassung im Seitenmodus eine Dateneingabe in der Editiermaske erst zu bestaetigen (z.B. durch Druecken der <RETURN>-Taste), ehe zum naechsten Feld uebergegangen wird
	OFF	automatischer Sprung zum naechsten Feld, wenn bei der Eingabe die Feldgrenze erreicht ist
CONSOLE	ON	Ausgabe der Ergebnisse auf dem Bildschirm
	OFF	Unterdrueckung von Anzeigen auf dem Bildschirm

DEBUG	ON	Ausgabe der mit SET ECHO bzw. SET STEP angeforderten Anzeigen auf dem Drucker, damit die Ausgaben im Seitenmodus nicht unterbrochen werden
	OFF	Unterdruecken der o.a. Ausgaben auf dem Drucker
DELETED	ON	Mit der Loeschmarkierung '*' versehene Saetze werden nicht angezeigt (z.B. bei LIST).
	OFF	Einbeziehung der mit '*' markierten Saetze (ausser bei COPY- und APPEND-Befehlen)
ECHO	ON	Anzeigen aller ausgefuehrten Befehle eines Programmes bei Ausfuehrung einer Befehlsdatei (s. Befehl DO) auf dem Bildschirm
	OFF	keine Anzeige der bei Ausfuehrung einer Befehlsdatei durchlaufenen Befehle
EJECT	ON	Seitenvorschub vor Druckbeginn einer mit dem REPORT-Befehl erstellten Liste
	OFF	Unterdruecken des Seitenvorschubs vor dem Druck einer Liste
ESCAPE	ON	Wird die <ESCAPE>-Taste gedrueckt, wird die Ausfuehrung einer Befehlsdatei unterbrochen.
	OFF	Unterbinden der Funktion der <ESCAPE>-Taste
EXACT	ON	Zwang zur Eingabe des vollstaendigen Suchbegriffes bei FIND-Befehlen und in Ausdruecken, d.h. alle alphanumerischen Werte muessen einschliesslich Leerstellen am Ende genau uebereinstimmen
	OFF	Bei Vergleichen (z.B. mit Schluesselwert im FIND-Befehl) von zwei Werten ist es ausreichend, wenn Uebereinstimmung des kuerzeren Wertes mit dem ersten Teil des laengeren Wertes besteht, z.B. "ABCDE" = "ABC".
INTENSITY	ON	In Abhaengigkeit von den Geraetemoeglichkeiten koennen auf dem Bildschirm im Seitenmodus mit unterschiedlichen Helligkeitsstufen oder invertierten Darstellungen Markierungen vorgenommen werden.
	OFF	Unterdrueckung der unterschiedlichen Helligkeitsstufen
LINKAGE	ON	Primaer- und Sekundaerdatei koennen parallel verarbeitet werden, indem alle diesem SET-Befehl folgenden Befehle, die mit einem <bereich> arbeiten (wie z.B. LIST, REPORT, SUM), sowohl in der Primaer- als auch in der Sekundaerdatei auf den gleichen <bereich> positioniert werden. Der SET-Befehl muss vor dem USE gegeben werden.
	OFF	Primaer- und Sekundaerdatei sind unabhaengig voneinander positioniert.

PRINT	ON	Protokollierung von Ergebnissen auf dem Drucker
	OFF	keine Protokollierung auf dem Drucker
RAW	ON	lueckenloses Auflisten der Felder bei den Befehlen DISPLAY, LIST, ?
	OFF	Einfuegen von Leerzeichen zwischen den Feldern bei DISPLAY, LIST, ? Ein Semikolon um Datenfeld bewirkt Zeilenumbruch.
SCREEN	ON	Umschalten auf Seitenmodus fuer die Befehle APPEND, INSERT, EDIT, CREATE, READ
	OFF	Umschalten vom Seitenmodus auf zeilenweise Eingabe
STEP	ON	Bei Ausfuehren einer Befehlsdatei haelt REDABAS nach Beenden jedes Befehls an und wartet auf die Eingabe des Nutzers. Von REDABAS wird folgender Text ausgegeben: Abarbeitung? (J=Befehl, N=Befehl von Konsole, ESC=Abbruch). Der Nutzer kann durch Eingabe von J, N, ESC entscheiden, ob er zusaetzliche Befehle von der Konsole eingeben will (N), die Ausfuehrung der Befehlsdatei abbrechen will (ESC) oder die Ausfuehrung des naechsten Befehls in der Datei veranlassen will (J).
	OFF	keine Unterbrechung bei Ausfuehrung einer Befehlsdatei (wenn nicht dort entsprechend programmiert)
TALK	ON	Anzeigen der Ergebnisse von Befehlen (z.B. Datensatznummer bei gefundenem Satz)
	OFF	kein Anzeigen der Ergebnisse von Befehlen

Bei Form b) des SET-Befehls gibt es folgende Moeglichkeiten:

SET ALTERNATE TO <dateiname>

Dieser Befehl steht in Verbindung mit SET ALTERNATE ON. Es wird eine mit <dateiname> bezeichnete Datei eroeffnet, in die alle Ein- und Ausgaben geschrieben werden, die auch auf dem Bildschirm erscheinen. Fehlt in der Angabe der Dateityp, wird er von REDABAS mit ".TXT" ergaenzt. Gibt es bereits eine Datei mit gleichem <dateinamen>, wird sie ueberschrieben.

Erst, wenn SET ALTERNATE ON gegeben wurde, beginnt die Protokollierung in die Datei. Protokolle lassen sich auch stueckweise aufbauen, indem mit ON bzw. OFF ein- bzw. ausgeschaltet wird.

Beispiel:
```
SET ALTERNATE TO protdat
  (Befehlsfolge)
SET ALTERNATE ON
  (Fortsetzung der Befehlsfolge, die ab jetzt in protdat.TXT
  geschrieben wird)
SET ALTERNATE OFF
  (Befehlsfolge ohne Protokollierung)
SET ALTERNATE ON
```

(Befehlsfolge, die ab jetzt wieder in protdat.TXT
geschrieben wird)
SET ALTERNATE TO prot2
(Befehlsfolge, ab jetzt in Datei prot2.TXT protokolliert)

SET CALL TO <adresse>

Mit diesem Befehl wird eine <adresse> im Arbeitsspeicher angegeben, zu der bei einem folgenden CALL-Befehl verzweigt werden soll. Die <adresse> ist dezimal anzugeben und sollte nicht unter 42752 (hex. A700) liegen, da diese Bereiche von REDABAS belegt werden. Nur der SORT-Befehl kann Bereiche benutzen, die oberhalb dieser Adresse liegen.

SET DATE TO <datum>

Dieser Befehl erlaubt die Eingabe des aktuellen Datums auch waehrend der Arbeit mit REDABAS.

SET DEFAULT TO <laufwerk>

Damit wird das hier benannte <laufwerk> fuer vom Nutzer anzugebende Dateien (nicht REDABAS-Systemdateien) bestimmt.

Ist bei Dateinamen das Laufwerk nicht explizit angegeben, ergaenzt REDABAS intern die Angabe mit dem hier definierten <laufwerk>. Beim Start von REDABAS ist das zunaechst das Laufwerk mit den Systemdateien.

<laufwerk> kann mit oder ohne Doppelpunkt angegeben werden. Man kann auch eine temporaere Speichervariable mit vorangestelltem &(Makro) verwenden.

Der SET-Befehl aendert nicht die Voreinstellung des Betriebssystems, er wirkt nur in REDABAS.

SET FORMAT TO [SCREEN]
[PRINT]
[<maskendatei>]

Mit den ersten beiden Moeglichkeiten dieses Befehls wird festgelegt, ob die mit dem @-Befehl erzeugten Ausgaben auf dem Bildschirm oder Drucker erfolgen sollen. Die dritte Variante veranlasst, dass bei den Befehlen READ, APPEND, INSERT, EDIT oder CREATE die <maskendatei> gelesen wird und die dort enthaltenen @-Befehle ausgefuehrt werden. Bei aktivierter Maskendatei kann ein Datensatz nicht durch die Tastenkombination <CONTROL-L> geloescht werden.

SET FORMAT TO ohne Option setzt die vorangegangene Zuweisung ausser Kraft.

SET HEADING TO <text>

Mit diesem Befehl wird REDABAS-intern der <text> (mit einer max. Laenge von 60 Zeichen) abgespeichert und bei mit REPORT erstellten Listen als Kopfzeile gedruckt. Fuer <text> kann auch eine Variable in Form der Makroersetzung angegeben werden.

SET INDEX TO [<indexdatei1>[,<indexdatei2>,...]]

Dieser Befehl dient zum gleichzeitigen Aktualisieren von bis zu sieben Indexdateien. Waren bis zu diesem Befehl schon Indexdateien aktiviert, werden sie damit geschlossen. Mit dem Aufruf einer neuen Indexdatei wird der Datensatzzeiger in der Datenbankdatei nicht veraendert, waehrend die Position in der Indexdatei noch undefiniert ist. Bei allen Befehlen, die mit NEXT -ausgehend von der augenblicklichen Position- auf davorliegende oder nachfolgende Saetze uebergehen, muss zunaechst FIND oder GOTO gegeben werden, um Indexdatei und Datenbankdatei gleichzuschalten.

Bei mehreren Indexdateien ist die zuerst genannte die Hauptindexdatei, mit der die Suchoperationen ausgefuehrt werden und deren Sortierordnung der Nutzer sieht. Alle weiteren aufgefuehrten Indexdateien sind nicht sichtbar, werden aber mit APPEND, EDIT usw. auch aktualisiert.
Mit SET INDEX TO ohne Angabe von <dateinamen> werden alle aktivierten Indexdateien freigegeben und im Verlauf der weiteren Arbeit wird nicht mehr auf Indexdateien zugegriffen.

SET MARGIN TO <zahl>

Mit diesem Befehl wird der linke Rand einer Liste (Druckausgabe) auf die Position von <zahl> positioniert.

SET <taste> TO '<befehl>;'

Dieser Befehl gilt nur fuer REDABAS unter SCP1700.
Durch diesen Befehl kann der Nutzer den Funktionstasten der Tastatur (PF1 bis PF10) einen REDABAS-Befehl zuordnen und somit bei haeufig benutzten Befehlen Eingabeaufwand sparen.
Der <befehl> ist mit Semikolon abzuschliessen und in Hochkommas einzusohliessen.
Fuer <taste> ist eine der symbolischen Bezeichnungen F1 bis F10 anzugeben. Beim Start von REDABAS gilt die Voreinstellung fuer die Tastenbelegung.
Im Betriebssystem SCP kann die Zuweisung von Funktionstasten nur ausserhalb von REDABAS erfolgen (z.B. beim PC1715 durch das Installationsprogramm INSTSCP).

SKIP SKIP

SKIP [+][-<numerischer ausdruck>]

Mit SKIP kann man die aktuelle Position des Datensatzzeigers veraendern. Der Zeiger weist immer auf den Datensatz, auf den gerade zugegriffen wird. Die neue Position wird durch Addition oder Subtraktion des Wertes von <ausdruck> erreicht. <ausdruck> kann direkt als Zahl angegeben werden, oder eine Zahl ergibt sich durch Auswertung von <ausdruck>.

Bei indizierten Dateien wird die Position des Datensatzzeigers von der Reihenfolge der Indexdatei bestimmt. Es ist also zu unterscheiden, ob eine Datenbankdatei mit oder ohne Indizierung verwendet wird.

Beispiel:
SKIP -5

SORT ON <feldname> TO <dateiname> [ASCENDING]
[DESCENDING]

Mit diesem Befehl wird die eroeffnete Datei nach dem mit <feldname> bezeichneten Datenfeld sortiert und in die mit <dateiname> benannte Ausgabedatei gespeichert. Die eroeffnete Datei wird nicht veraendert, sie bleibt weiterhin im Zugriff. Die neue, sortierte Datei muss mit USE fuer die Bearbeitung aktiviert werden.

Wurde die Datei zuvor mit APPEND erweitert, muss die Datei durch einen USE-Befehl abgeschlossen und wieder eroeffnet werden.

Es ist zu beachten, dass in einem SORT-Befehl nur nach einem Datenfeld sortiert werden kann. Will man nach mehreren Datenfeldern sortieren, muss man den Befehl SORT mit den weiteren <feldnamen> wiederholen und erhaelt so eine stufenweise Sortierung. Dabei wird so vorgegangen, dass der erste Sortiervorgang mit dem letzten Sortierfeld begonnen wird und so schrittweise weiter bis zum ersten Datenfeld, dem Hauptsortierkriterium, gearbeitet wird.

Beim Vergleich der Datenfeldwerte erfolgt die Darstellung im ASCII-Zeichensatz. Damit ist eine aufsteigende Reihenfolge (ASCENDING) gegeben ("A" vor "B", Ziffern vor Buchstaben usw.). Ohne Angabe der Sortierfolge im Befehl wird also seitens REDABAS immer ASCENDING angenommen. Soll absteigend sortiert werden, muss DESCENDING angegeben werden.

Es sei an dieser Stelle auf den Befehl INDEX verwiesen, der prinzipiell auf die gleiche Wirkung ausgerichtet ist, jedoch anders arbeitet. Abschnitt 3.4. dieser Programmtechnischen Beschreibung geht ausfuehrlich auf beide Befehle ein.

Beispiel:

SORT ON teil TO teilsort DESCENDING

(und s. Abschnitt 3.4.)

STORE STORE

STORE <ausdruck> TO <speichervariablen>

STORE speichert den Inhalt von <ausdruck> bzw. dessen Ergebnis (z.B. wenn <ausdruck> eine Formel ist) in einer oder mehreren temporaeren <speichervariablen>. Existiert die <speichervariable> noch nicht, wird sie mit diesem Befehl angelegt.

Eine <speichervariable> bleibt nur fuer die Dauer einer REDABAS-Sitzung erhalten und kann waehrend dieser Zeit beliebig oft aufgerufen oder geaendert werden. Mit SAVE kann man die <speichervariable> in eine Datei abspeichern und sie so ueber eine REDABAS-Sitzung hinaus erhalten. Mit RESTORE

wird diese Datei wieder verfuegbar gemacht. RELEASE loescht eine oder mehrere Variable.

Mit STORE kann auch der Datentyp einer existierenden <speichervariablen> wieder geaendert werden (z.B. mit den Funktionen VAL oder STR). Jedoch kann dann als Quelle (<ausdruck>) und Ziel nicht dieselbe Speichervariable genannt werden.

Beispiel:

```
STORE 'wert' TO Swert
```

(und s. Abschnitt 3.5.)

SUM SUM

```
SUM<ausdruck1> [,<ausdruck2>,...][TO <speichervariable1>
   [,<speichervariable2>,...]] [<bereich>][FOR <bedingung>]
```

Mit SUM werden numerische Datenfelder bzw. aus diesen gebildete <ausdruecke> der aktivierten Datei addiert. Ist weder <bereich> noch <bedingung> angegeben, wird die gesamte Datei in die Summierung einbezogen. Sind die entsprechenden Parameter spezifiziert, werden nur die Saetze aus dem <bereich> und die der <bedingung> genuegenden Saetze beruecksichtigt.

Mit der Option TO kann das Ergebnis der Summierung in eine <speichervariable> geschrieben werden (ansonsten wird es nur angezeigt). Auch wenn die TO-Option nicht benutzt wird, legt REDABAS fuer die Zwischensummen Speichervariablen an, die nach Abschluss des SUM-Befehls automatisch geloescht werden. Dabei ist es moeglich, gleichzeitig 5 Datenfelder bzw. <ausdruecke> von Satz zu Satz zu addieren und jedes Ergebnis fuer sich in eine <speichervariable> abzuspeichern (Summe von <ausdruck1> in <speichervariable1> usw.).

<bereich> erfasst nur die Saetze, die nicht fuer eine Loeschung markiert sind.

Beispiel:

```
SUM betrag FOR .NOT. beza TO Soffen
```

(und s. Abschnitt 3.5.1.)

TEXT
<text>
ENDTEXT

Mit dem Befehl TEXT koennen bei der Ausfuehrung von Befehlsdateien ein oder mehrere Kommentarzeilen ausgegeben werden. Die Textzeilen sind durch ENDTEXT abzuschliessen. Es ist zu beachten, dass TEXT und ENDTEXT auf separaten Zeilen stehen muessen.

TOTAL ON <schluessel> TO <dateiname> [FIELD <feldnamen>]
[FOR <bedingung>]

Der Befehl TOTAL dient zum Komprimieren von Dateien, d.h. es werden diejenigen Datensaetze zusammengefasst, die in dem mit <schluessel> bezeichneten Datenfeld den gleichen Inhalt haben. Dabei wird der Inhalt der numerischen Felder addiert. Die neue komprimierte Datei wird in der mit <dateiname" angegebenen Datei abgespeichert.

Voraussetzung fuer die Anwendung von TOTAL ist, dass die aktivierte Datei nach dem <schluessel> sortiert oder indiziert ist.

Wird die Struktur der mit <dateiname> bezeichneten Zieldatei vor dem Befehl TOTAL nicht festgelegt, wird die Struktur der aktivierten Datei uebernommen. Diese Datei bleibt nach Ausfuehrung von TOTAL voll erhalten.

Ist im Befehl TOTAL die Option FIELD angegeben, werden nur die unter <feldnamen> aufgefuehrten numerischen Felder summiert. Ohne diese Option werden alle numerischen Felder addiert (wenn noch keine Zieldatei existiert) bzw. die Inhalte aller in beiden Dateien enthaltenen numerischen Felder addiert.

Es ist darauf zu achten, dass die zu addierenden numerischen Felder ausreichend dimensioniert sind, um die bei der Komprimierung entstehenden Summen aufnehmen zu koennen. Besser ist, die Struktur der Zieldatei vorher mit ausreichender Dimensionierung zu erzeugen.

Bei der Option FIELD koennen aber unter <feldnamen> auch nicht-numerische Felder angegeben werden. Diese werden zwar logischerweise nicht summiert, da aber zu jedem <schluessel>-Wert in der Zieldatei nur ein Datensatz angelegt wird, kann man auf diese Weise doppelte Saetze eliminieren.

Mit der FOR-Klausel kann die Summierung auf die Datensaetze der aktivierten Datei eingeschraenkt werden, die die <bedingung> erfuellen.

Beispiel:

TOTAL ON ort TO datort
(und s. Abschnitt 3.5.4.)

UPDATE FROM <dateiname> ON <schluessel> [ADD <feldnamen>]
[REPLACE <feldnamen> [WITH <feldnamen>]] [RANDOM]

Der Befehl UPDATE aktualisiert die mit USE eroeffnete Datei, indem Daten dieser Datei durch den Inhalt der mit <dateiname> bezeichneten Datei ergaenzt bzw. ersetzt werden. Die Beziehungen zwischen den zusammengehoerigen Datensaetzen beider Dateien werden ueber den gleichen Inhalt von <schluessel> hergestellt.

Zur effektiven Abarbeitung des UPDATE-Befehls muss eine der drei nachfolgenden Voraussetzungen erfuellt sein:
- beide Dateien liegen nach dem <schluessel> sortiert vor
- die aktivierte Datei ist nach dem <schluessel> indiziert und die zweite Datei sortiert
- nur die aktivierte Datei ist nach dem <schluessel> indiziert, die zweite Datei ist unsortiert und die RANDOM-Klausel wird gesetzt.

Mit ADD <feldnamen> werden die Inhalte der dort spezifizierten Felder zu den Feldinhalten der eroeffneten Datei hinzugefuegt. Mit REPLACE <feldnamen> werden die Inhalte der genannten Felder in der eroeffneten Datei mit den Daten aus der "FROM-Datei" ueberschrieben.

Beim Ersetzen muessen entweder die Namen der betreffenden Felder in beiden Dateien uebereinstimmen oder dem <feldnamen> in der REPLACE-Klausel wird ein <feldname> in der WITH-Klausel zugeordnet.

Eine Ergaenzung und/oder Ersetzung (ADD und REPLACE koennen gleichzeitig in einem Befehl benutzt werden) erfolgt dann, wenn die Inhalte des als <schluessel> definierten Datenfeldes in beiden Dateien uebereinstimmen.

Das Aktualisieren selbst geschieht auf folgende Weise, wenn beide Dateien geordnet sind:
Es werden beide Dateien auf den Anfang positioniert. Aus jeder Datei wird ein Datensatz gelesen. Stimmen die Inhalte der <schluessel> ueberein, wird aktualisiert. Gibt es keine Uebereinstimmung, bleibt der aktuelle Datensatz unveraendert und aus der Datei, dessen Schluesselinhalt "kleiner" war, wird der naechste Satz gelesen. Dies wird so lange fortgesetzt, bis eine der beiden Dateien ihr Ende erreicht hat. Durch die geforderte aufsteigende Sortierung beider Dateien ist gewaehrleistet, dass auf diese Weise alle Aenderungen realisiert werden.

Wenn RANDOM angegeben ist, wird die zweite Datei sequentiell abgearbeitet und der zugehoerige Datensatz aus der aktivierten Datenbank ueber den Indexzugriff verfuegbar gemacht.

USE USE

USE [<dateiname> [INDEX <indexdatei1>,<indexdatei2>,...]]

Mit USE wird eine bereits existierende Datenbankdatei (Dateityp .DBD) fuer die Bearbeitung eroeffnet. War vor diesem USE-Befehl schon eine Datei eroeffnet, wird sie geschlossen und ist damit nicht mehr im Zugriff von REDABAS. Zu diesem Zeitpunkt werden seitens REDABAS statistische Daten aktualisiert, wie z.B. Datum der letzten Aktualisierung (s. Befehl DISPLAY STRUCTURE) usw.

USE ohne Angabe von <dateiname> schliesst die gerade aktivierte Datei im zugewiesenen Dateibereich.

Mit dem Zusatz INDEX wird eine zuvor durch den INDEX-Befehl indizierte Datei gemeinsam mit der/den hier aufgefuehrten <indexdatei(en)> aktiviert. Dies ermoeglicht den Zugriff auf die Datensaetze in der Reihenfolge des Schluessels (d.h. indexsequentiell) und die Benutzung entsprechender Befehle, wie z.B. FIND.

Mit einem USE-Befehl koennen bis zu sieben Indexdateien aktiviert werden. Dabei gilt der Schluessel der ersten Indexdatei als Hauptschluessel. Er ist die Basis fuer alle FIND-Befehle und auch der Befehl SKIP bezieht sich auf ihn als Ordnungskriterium. Diese <indexdateien> werden automatisch aktualisiert, wenn durch die Befehle APPEND, INSERT, PACK, BROWSE, EDIT, REPLACE, READ Datensaetze hinzugefuegt, geloescht oder Schluesselfelder geaendert werden.

Beispiel:

USE kunden INDEX ikunr

(und s. Abschnitt 3.1.1. und 3.4.)

WAIT WAIT

WAIT [TO <speichervariable>]

WAIT veranlasst REDABAS, das laufende Programm anzuhalten und so lange in dem Wartezustand zu verbleiben, bis ein Tastendruck durch den Bediener erfolgt. Auf dem Bildschirm erscheint die Nachricht FORTSETZEN MIT TASTE ...

Enthaelt der WAIT-Befehl eine TO-Klausel, wird das ueber die Tastatur eingegebene Zeichen in der angegeben temporaeren <speichervariablen> gespeichert. Diese TO-Option kann dann sehr vorteilhaft sein, wenn die Fortfuehrung einer Befehlsdatei in Abhaengigkeit von der Eingabe eines einzigen Zeichens erfolgt. Dies wird beispielsweise bei Menues angewandt.

Ein nicht sichtbares Steuerzeichen in einem WAIT-Befehl mit TO-Klausel (wie z.B. <RETURN>) hinterlegt in der <speichervariablen> ein Leerzeichen.

Bei dem Befehl WAIT ist es im Unterschied zu den Befehlen ACCEPT und INPUT nicht notwendig, eine Eingabe mit der <RETURN>-Taste abzuschliessen.

Beispiel:

```
WAIT TO Svar
```

(und s. Abschnitt 4.2.)

9. REDABAS-Schnittstellen

Mit dem einheitlichen und durchgaengigen Sprachkonzept von REDABAS kann eine Vielzahl von Datenverwaltungs- und Auswertungsproblemen geloest werden.

Fuer die Kommunikation zwischen REDABAS und anderen Programmen verfuegt REDABAS ueber zwei Schnittstellen, eine auf Datenebene und eine weitere auf Programmebene.

9.1. Daten-Schnittstelle

REDABAS ist in der Lage, Datenbankkopien in einem Format zu erzeugen, das auch von Anwendungsprogrammen verarbeitet werden kann, die in anderen Programmiersprachen (wie z.B. PASCAL, BASIC, FORTRAN) geschrieben worden sind.

Ebenso kann REDABAS Dateien lesen, die durch solche Programme erstellt wurden.

Das Format dieser Dateien entspricht den SCP-Konventionen fuer Textdateien, d.h. der Dateninhalt besteht aus Zeichen des ASCII-Alphabetes und jeder Datensatz ist durch <RETURN> und Zeilenvorschub (hexadezimal '0D0A') begrenzt. Die Datei muss durch <CONTROL-Z> (hexadezimal '1A') abgeschlossen sein. Die Datensaetze sind sequentiell organisiert und koennen konstant oder variabel lang sein.

Ausgabe

REDABAS erzeugt ueber den COPY- Befehl mit der Option SDF eine Textdatei, in der die Felder unterschiedlich begrenzt sein koennen (s. Abschnitt 8.2.). Das Ausgabeformat der Saetze haengt davon ab, welche Optionen im COPY-Befehl angegeben werden:

- Angabe von COPY...SDF
  Die Felder werden in voller, definierter Laenge ohne Begrenzungs- und Trennzeichen in der Textdatei angeordnet. Leerstellen werden nicht entfernt.

- Angabe von COPY...SDF DELIMITED
  Alle Felder im Ausgabesatz sind durch Kommas voneinander getrennt. Zeichenreihen sind in Hochkommas eingeschlossen, Leerstellen am Ende von Zeichenreihen-Feldern werden entfernt.

- Angabe von COPY...SDF DELIMITED WITH ,
  Fuehrende Leerstellen in numerischen Feldern und Leerstellen am Ende von Zeichenreihen-Feldern werden entfernt. Die Felder sind durch das in der WITH-Klausel angegebene Komma voneinander getrennt. Die Felder enthalten keine zusaetzlichen Begrenzungszeichen.

Eingabe

REDABAS verarbeitet Textdateien, die von anderen Programmen erzeugt wurden, mit dem APPEND-Befehl. Die Reihenfolge der Felder in der Textdatei muss der definierten Feldanordnung in der aktuellen Datenbankdatei entsprechen, an die die Saetze der Textdatei angehaengt werden. Analog zum COPY-Befehl koennen die Datenfelder unterschiedlich begrenzt sein:

- Angabe von APPEND...SDF
  Die Felder in der Textdatei werden in der gleichen Laenge erwartet, wie sie in der aktuellen Datenbankdatei definiert sind. Die Datenfelder muessen in der Textdatei ohne Trenn- und Begrenzungszeichen lueckenlos angeordnet sein.
- Angabe von APPEND ...SDF DELIMITED
  REDABAS erwartet die Datenfelder in der Textdatei durch Kommas voneinander getrennt. Felder vom Typ Zeichenreihe koennen in Hochkommas oder Anfuehrungszeichen eingeschlossen sein. Andere Begrenzungszeichen werden zum Feldinhalt gehoerig betrachtet. Felder, die groesser als die definierte Laenge in der Datenbankdatei sind, werden abgeschnitten, kuerzere Felder werden mit Leerzeichen aufgefuellt.

9.2. Programm-Schnittstelle

Waehrend einer REDABAS-Sitzung kann durch den CALL-Befehl eine Assemblerprozedur aufgerufen werden. Die Startadresse solch einer Prozedur ist vorher durch einen SET CALL TO <adresse>-Befehl mitzuteilen. Im Prozeduraufruf kann die Adresse einer temporaeren Speichervariablen uebergeben werden.

Unter SCP steht fuer die Assemblerprozeduren der Speicherbereich ab Adresse
   A400H (dezimal 41984) fuer REDABAS Version 1.0 bzw
   A700H (dezimal 42752) fuer REDABAS Version 2.0 bzw.
   AC00 (dezimal 44032) fuer REDABAS Version 3.0
bis zum Ende der Transient Program Area (TPA) einschliesslich des CCP-Bereiches zur Verfuegung.
Bei der Arbeit mit REDABAS Version 5.0 (unter SCP1700 kann der Speicherbereich ab D0C0H (dezimal 53440) innerhalb des aktuellen Codesegments fuer Assemblerprozeduren genutzt werden.
Wenn eine Assemblerprozedur geladen ist, kann kein SORT-Befehl benutzt werden, da dieser auch die obengenannten Speicherbereiche verwendet.
Bei der Erstellung einer Assemblerprozedur wird dem Programmverbinder die gewuenschte Ladeadresse mitgeteilt. Der Programmverbinder setzt der Prozedur eine Verschieberoutine voran. Das verbundene Programm wird vor dem REDABAS-Aufruf geladen. Dabei wird die Verschieberoutine abgearbeitet, die die eigentliche Problemprozedur auf die vorgesehene Startadresse im Speicher transportiert. Beim Start von REDABAS befindet sich bei dieser Arbeitsweise die durch den CALL-Befehl aufzurufende Problemprozedur bereits im Speicher.
Liegt die Assemblerprozedur im ASCII-HEX-Format vor, kann das Laden dieser Datei auch waehrend einer REDABAS-Sitzung durch den Befehl LOAD erfolgen ab (Version 3.0).

ABARBEITUNG DER BEFEHLSDATEI ABGEBROCHEN!
Fehler bei Abarbeitung eines Kommandos in der Befehlsdatei (z.B. Unterprogramm nicht vorhanden).

ANZAHL DER TEMPORAEREN VARIABLEN ZU GROSS (MAX.64)!
Maximal 64 temporaere Variablen erlaubt, nicht benoetigte Variablen loeschen!

AUSDRUCK FUER SUMMENBILDUNG FEHLT!
SUM erfordert Angabe der zu summierenden Felder.

AUSDRUCK IST NICHT NUMERISCH!
SUM erfordert einen numerischen Ausdruck fuer die Summierung.

BEFEHLSDATEI NICHT VORHANDEN!
Dateinamen oder Laufwerksangabe im DO-Kommando pruefen!

DATEI IST BEREITS ANDERWEITIG AKTIVIERT!
Datei kann nur in einem Bereich eroeffnet sein. Aktive Dateien mit USE oder CLEAR schliessen!

DATEI KANN NICHT EROEFFNET WERDEN!
Pruefen, ob Datei vorhanden ist!

DATEI NICHT VORHANDEN!
Pruefen, ob Datei vorhanden ist, z.B. mit DISPLAY FILES...

DATEIENDE ERREICHT!
Es wurde kein (weiterer) Datensatz entsprechend der LOCATE-Bedingung gefunden.

DATEIVERZEICHNIS DER DISKETTE IST VOLL!
Nicht benoetigte Dateien loeschen!

DATENBANKDATEI WURDE OHNE INDEXDATEI AKTIVIERT!
FIND ist nur auf indizierte Dateien moeglich, Index-Datei mit USE anschliessen!

DATENFELD NICHT VORHANDEN!
Pruefen, ob Datenfeld in Struktur vorhanden bzw. ob Feldname korrekt ist! REPLACE neu eingeben!

DISKETTE IST VOLL!
Loeschen der nicht benoetigten Dateien!

***DIVISION DURCH NULL!
Division durch Null ist nicht erlaubt!

EINFUEGEN NICHT MOEGLICH, DATEI LEER BZW. DATEIENDE ERREICHT!
INSERT durch APPEND ersetzen!

ENDE DES SUCHBEREICHES!
Bei LOCATE mit <bereich>-Angabe (NEXT) wurde kein Datensatz entsprechend der <bedingung> gefunden!

"FIELD"-ANGABE FEHLT!
CHANGE-Befehl neu eingeben!

"FOR"-ANGABE FEHLT!
JOIN-Befehl neu mit "FOR"-Klausel eingeben!

FORMAT-DATEI KANN NICHT EROEFFNET WERDEN!
Disketteninhalt ueberpruefen!

"FROM"-ANGABE FEHLT!
UPDATE-Befehl neu mit "FROM"-Angabe eingeben!

INDEXDATEI GEHOERT NICHT ZUR DATENBANKDATEI!
Nur zur Datenbankdatei gehoerende Index-Dateien verwenden!

JOIN ERZEUGT MEHR ALS 65534 SAETZE!
"FOR"-Klausel einengen!

KEINE DATENBANKDATEI AKTIV! GEBEN SIE DEN DATEINAMEN EIN!

KEINE REDABAS-DATENBANKDATEI!
Die eroeffnete Datei ist nicht durch REDABAS aufgebaut worden.

MAKRO IST KEINE ZEICHENKETTE!
Makroexpansion verlangt Variablen mit Zeichenreihe als Wert.

MAKRO NICHT VORHANDEN!
Dem Zeichen fuer Makroersetzung (&) muss der Name einer temporaeren Speichervariablen folgen.

MAX.SCHACHTELUNGSTIEFE UEBERSCHRITTEN!
Befehlsdateien ueberarbeiten und zusammenfassen! Summe eroeffneter Befehlsdateien und anderer Dateien max. 16!

MEHR ALS 5 SUMMANDEN ANGEGEBEN!
Sum-Befehl nur mit max. 5 Feldern moeglich.

"ON"-ANGABE FEHLT!
UPDATE- oder INDEX-Befehl neu mit "ON"-Angabe eingeben!

PROGRAMM RDBASOVR.COM NICHT GEFUNDEN!
REDABAS vom aktuellen Laufwerk starten!

SATZLAENGE UEBERSCHREITET MAXIMUM VON 1000 BYTES
Feldlaenge oder Feldanzahl kuerzen!

SCHLUESSEL IM INDEX NICHT VORHANDEN!
Indexdatei nicht aktuell, Datenbankdatei reinindizieren!

SCHLUESSEL ODER ZEICHENKETTE NICHT VORHANDEN!

SPEICHERBEREICH FUER TEMPORAERE VARIABLE IST VOLL!
Nicht benoetigte temporaere Variablen loeschen! Der Speicherbereich darf 1536 Bytes (einschliesslich Variablennamen und Attribute) nicht ueberschreiten.

SYNTAXFEHLER, WIEDERHOLEN SIE DIE EINGABE!
Die Kommandozeile ist syntaktisch falsch. Das '?' weist auf die vermutlich falsche Angabe.

SYNTAXFEHLER IN FORMAT-DEFINITION!
@ SAY GET PICTURE - Befehl ist fehlerhaft, neu eingeben!

SYSTEMFEHLER BEI EIN- ODER AUSGABE!

"TO"-ANGABE FEHLT!
Befehl mit "TO"-Angabe neu eingeben!

*** UEBERLAUF IM NUMERISCHEN FELD!

UNBEKANNTES KOMMANDO!
Kommando ueberpruefen und neu eingeben!

UNERWARTET DATEIENDE ERREICHT!
Indexdatei passt nicht zur aktivierten Datei, eventuell neu indizieren!

UNGUELTIGE ANGABE BEI "GOTO"!
Satz-Nr. muss im Bereich zwischen 1 und 65535 und im belegten Bereich der Datei liegen.

UNGUELTIGER NAME FUER EINE VARIABLE!
Felder und Variablen duerfen nur alphanumerische Zeichen und Doppelpunkte enthalten.

UNTERSCHIEDLICHER DATENTYP BEI QUELLE UND ZIEL!
Datentyp muss uebereinstimmen!

UNTERSCHIEDLICHE SCHLUESSELLAENGEN!
UPDATE-Befehl fordert uebereinstimmende Schluessellaengen.

UNZULAESSIGE ANZAHL VON DEZIMALSTELLEN!
Bei Felddefinition Anzahl Dezimalstellen neu eingeben!

UNZULAESSIGER DATEINAME!
Syntaxfehler im Dateinamen.

UNZULAESSIGE FELDLAENGE!
Die Feldlaenge darf minimal 1 und max. 254 Stellen betragen. Bei String-Operationen (Zeichenreihen) darf die Stellenangabe den Wert 254 nicht ueberschreiten.

UNZULAESSIGER FELDNAME!
Syntaxfehler im Feldnamen.

UNZULAESSIGER FELDTYP!
Zulaessiger Feldtyp ist C (alphanumerisch), N (numerisch) oder L (logisch).

*** UNZULAESSIGE ZEICHENKETTE!

VARIABLE NICHT VORHANDEN!
Variablennamen ueberpruefen bzw. temporaere Variable erzeugen!

"WITH"-ANGABE FEHLT!
REPLACE-Befehl mit "WITH"-Angabe neu eingeben!

ZEICHENFOLGE ZU LANG!
Feldlaenge bzw. max. Laenge der Befehlszeile ueberschritten!

ZEIGER WEIST AUF EINEN SATZ AUSSERHALB DES DATENBANKBEREICHES!

ZU VIELE DATEIEN AKTIVIERT!

Es duerfen max. 16 Dateien (aller Typen) gleichzeitig eroeffnet werden.
ZU VIELE INDEXDATEIEN ANGEGEBEN (MAX.7)!

ZU VIELE "RETURN" ANGEGEBEN!
Ueberpruefen der Struktur der Befehlsdateien bzgl. Schachtelung und Schleifen!

Abkuerzungsverzeichnis

ASCII	-	American Standart Code For Information Interchange
AC	-	Arbeitsplatzcomputer
BC	-	Buerocomputer
PC	-	Personalcomputer
PLZ	-	Postleitzahl
RAM	-	Random Access Memory
SCP	-	Single User Control Program

Sachwortverzeichnis

ACCEPT 79, 136
APPEND 22,41,96,136 ff., 185 f.
Arbeitsweise 11
Arithmetische Operationen 111 f.
Ausdruecke 110 ff.
Auswerten 48

Beenden 15 f.
Befehle 127 ff.
Befehlsdateien 103
Befehlsfolgen 75
BROWSE 42, 138 f.

CALL 140
CANCEL 141
CHANGE 26f, 141
Character-Funktion 119
CHR 119
CLEAR 82, 142
CONTINUE 35, 142
COPY 39, 142 ff., 185
COUNT 50, 145
CREATE 17, 145 f.
Cursor 12
Cursorsteuerung 106

DATE 119
Dateien 102
  Aendern 17 ff.
  Einrichten 17 ff.
  Erstellen 17 ff.
  Gleichzeitiges Bearbeiten 66 ff.
  Verknuepfen 72 ff.
Dateiende-Funktion 119 f.
Datei-Funktion 120
Dateistruktur 39 ff.
  Aendern 39 ff.
Datenbank 100 ff.
  Aufbau 100 ff.
Datenbankdateien 102
Datenfeld 17 ff.
  Abfragen 36
  als Suchbegriff 30 ff.
  Selektion 29

Datenorganisation 100 ff.
Datensatz 17
Aendern 23
bedingte Anzeige 36
bedingte Auflistung 30
bedingte Suche 35
Loeschen 27 ff.
Positionieren auf 36
Selektion 29
Struktur 17
Zaehlen 50
Datentyp 101
Datum-Funktion 119
DELETE 28, 147
Dialogfuehrung 105
DISPLAY 35, 147
DO 83, 84, 89, 149f.

EDIT 23, 151
EJECT 152
ELSE 89
ENDDO 84, 152
EOF 119
ERASE 76

Fehlerbehandlung 105
Feldausschnitt 32
Feldformat 101
FILE 120
FIND 46, 153 f.
Funktionen 115 ff.
der Datenbankverwaltung 17 ff.
Funktionsuebersicht 116

GO 36, 154
Grossbuchstabenfunktion 116

HELP 155
Hinweise, allgemeine 13 ff.
IF 89, 155
INDEX 45, 156
Indexdateien 103
Indizieren 45
INPUT 157
INSERT 23, 158
Installieren 11 ff.
INT 120
Integer-Funktion 120 f.

JOIN 72, 158 f.

Konstanten 110

Laenge-Funktion 121
Leistungsparameter 10
LEN 121
LIST 20, 159
Listengenerator 54
LOAD 159, 186

LOCATE 35, 160
Loeschabfragefunktion 118
Logische Operationen 113
LOOP 160

Makros 125 f.
Maske 19
fuer Datenerfassung 19
Maskendateien 104
MODIFY 35, 75, 77, 161

NOTE 162

Operationen 111
Rangfolge 114
ORD 121
Ordinalitaet-Funktion 121

PACK 162 f.
PEEK 122
POKE 163
Programmieren 75
Programmierte Dateneingabe 95
Programmschleifen 83
Programmverzweigungen 89

QUIT 163

RDBASINS.COM 11, 13
RDBASMSG.TXT 155
READ 97, 164
RECALL 165
REDABAS.COM 11, 13
REINDEX 165
RELEASE 165
REMARK 94, 166
RENAME 166
REPLACE 27, 167
REPORT 55, 168 ff.
Reportdateien 104
RESET 171
RESTORE 171 f.
RETURN 76, 97, 172

Satznummer-Funktion 117
SAVE 172
Schnittstellen 185 ff.
Daten- 185 f.
Programm- 186
SCP-Dateityp 17
Seitenmodus 12, 106
SELECT 67, 173
SET 78, 174 ff.
Sicherungsdateien 105
SKIP 68, 178
SORT 43, 179
Sortieren 43 ff.
Speicheranzeige-Funktion 122
Starten 15 f.
STORE 86, 179 f.

STR 122
String-Funktion 122
String-Operationen 114
Stringsuche-Funktion 118
Substring-Funktion 117
Suchbedingungen 33 ff.
Suchbegriff
    Datenfeld als 30 ff.
    Feldausschnitt als 32
SUM 49, 180
Summenbildung 49
Symboldefinitionen 129

Tastenkombinationen 26, 40, 76, 107, 108
TEST 123
Test-Funktion 123
TEXT 181
Textdateien 104
TOTAL 51, 181
TRIM 81, 123
Trimmfunktion 123
TYPE 124
Typ-Funktion 124

UPDATE 71, 182
USE 20, 183

VAL 124
Variablen 110
Variablendateien 103
Verdichten 51 ff.
Vergleichsoperationen 112
Verknuepfen
    von Dateien 72
    von Suchbedingungen 33

WAIT 87, 94, 183 ff.
Wert-Funktion 124
Wiederauffinden 29 ff.
Zaehlen 50

Abschrift erstellt

Elmar Klinder
Götz Hupe

AG 706/378/87 III/18/97